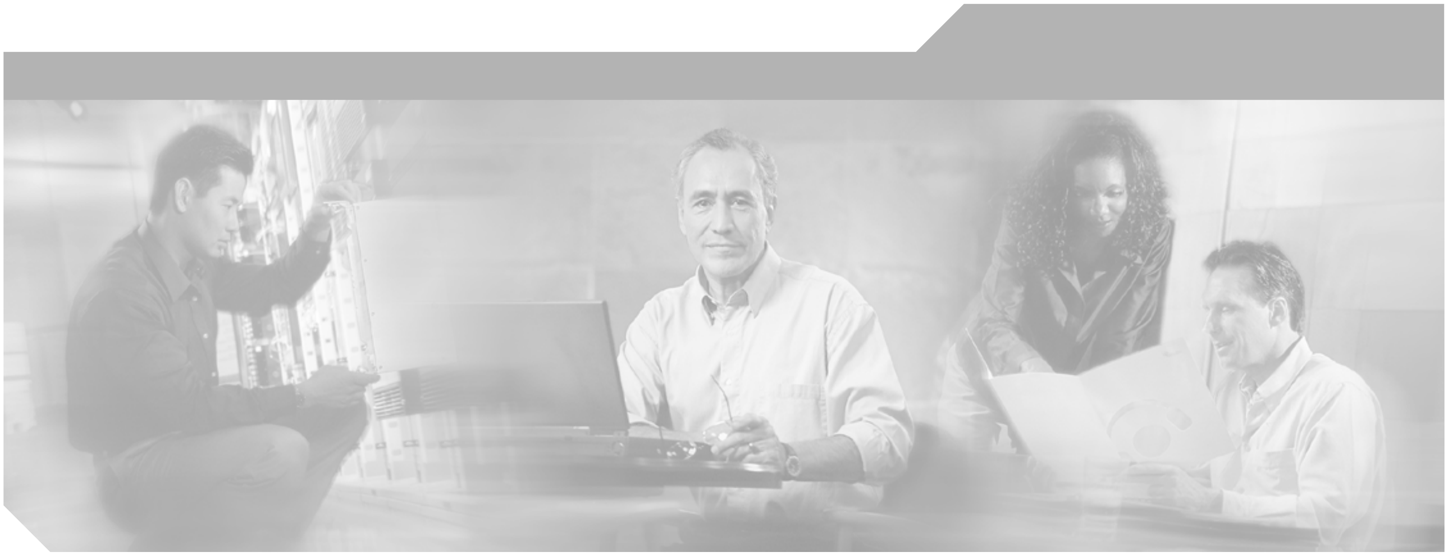


Cisco IOS XR Routing Configuration Guide

Cisco IOS XR Software Release 3.2

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Text Part Number: OL-5554-05

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StrataView Plus, TeleRouter, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0502R)

Cisco IOS XR Routing Configuration Guide

CONTENTS

Preface

This is the preface for the *Cisco IOS XR Routing Configuration Guide*.

The preface contains the following sections:

- Document Revision History, page xi
- Obtaining Documentation, page xii
- Documentation Feedback, page xiii
- Cisco Product Security Overview, page xiii
- Obtaining Technical Assistance, page xiv
- Obtaining Additional Publications and Information, page xv

Document Revision History

The Document Revision History table records technical changes to this document. Table 1 shows the document revision number for the change, the date of the change, and a brief summary of the change. Note that not all Cisco documents use a Document Revision History table.

Table 1 Document Revision History

Revision	Date	Change Summary
OL-5554-05	November 2005	Added description for the OSPFv3 Graceful Restart feature. Added descriptions for the multicast-intact option in IS-IS and OSPFv2.
OL-5554-04	August 31, 2005	Implementing IS-IS on Cisco IOS XR Software changes: Updated the IS-IS module to include the ability to configure a broadcast medium connecting two networking devices as a point-to-point link.
OL-5554-02	October 31, 2004	Implementing IS-IS on Cisco IOS XR Software changes: Changes to lsp-gen-interval, spf-interval, and show isis spf-log information.
OL-5554-01	July 30, 2004	Initial release of this document.

Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/techsupport

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

Product Documentation DVD

Cisco documentation and additional literature are available in the Product Documentation DVD package, which may have shipped with your product. The Product Documentation DVD is updated regularly and may be more current than printed documentation.

The Product Documentation DVD is a comprehensive library of technical product documentation on portable media. The DVD enables you to access multiple versions of hardware and software installation, configuration, and command guides for Cisco products and to view technical documentation in HTML. With the DVD, you have access to the same documentation that is found on the Cisco website without being connected to the Internet. Certain products also have .pdf versions of the documentation available.

The Product Documentation DVD is available as a single unit or as a subscription. Registered Cisco.com users (Cisco direct customers) can order a Product Documentation DVD (product number DOC-DOCDVD=) from Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Ordering Documentation

Beginning June 30, 2005, registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Nonregistered Cisco.com users can order technical documentation from 8:00 a.m. to 5:00 p.m. (0800 to 1700) PDT by calling 1 866 463-3487 in the United States and Canada, or elsewhere by calling 011 408 519-5055. You can also order documentation by e-mail at tech-doc-store-mkpl@external.cisco.com or by fax at 1 408 519-5001 in the United States and Canada, or elsewhere at 011 408 519-5001.

Documentation Feedback

You can rate and provide feedback about Cisco technical documents by completing the online feedback form that appears with the technical documents on Cisco.com.

You can send comments about Cisco documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Cisco Product Security Overview

Cisco provides a free online Security Vulnerability Policy portal at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

From this site, you can perform these tasks:

- Report security vulnerabilities in Cisco products.
- Obtain assistance with security incidents that involve Cisco products.
- Register to receive security information from Cisco.

A current list of security advisories and notices for Cisco products is available at this URL:

http://www.cisco.com/go/psirt

If you prefer to see advisories and notices as they are updated in real time, you can access a Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed from this URL:

http://www.cisco.com/en/US/products/products_psirt_rss_feed.html

Reporting Security Problems in Cisco Products

Cisco is committed to delivering secure products. We test our products internally before we release them, and we strive to correct all vulnerabilities quickly. If you think that you might have identified a vulnerability in a Cisco product, contact PSIRT:

- Emergencies—security-alert@cisco.com

  An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.

- Nonemergencies—psirt@cisco.com

In an emergency, you can also reach PSIRT by telephone:

- 1 877 228-7302
- 1 408 525-6532

Tip We encourage you to use Pretty Good Privacy (PGP) or a compatible product to encrypt any sensitive information that you send to Cisco. PSIRT can work from encrypted information that is compatible with PGP versions 2.*x* through 8.*x*.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one linked in the Contact Summary section of the Security Vulnerability Policy page at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

The link on this page has the current PGP key ID in use.

Obtaining Technical Assistance

Cisco Technical Support provides 24-hour-a-day award-winning technical assistance. The Cisco Technical Support & Documentation website on Cisco.com features extensive online support resources. In addition, if you have a valid Cisco service contract, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not have a valid Cisco service contract, contact your reseller.

Cisco Technical Support & Documentation Website

The Cisco Technical Support & Documentation website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support & Documentation website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

http://tools.cisco.com/RPF/register/register.do

Note Use the Cisco Product Identification (CPI) tool to locate your product serial number before submitting a web or phone request for service. You can access the CPI tool from the Cisco Technical Support & Documentation website by clicking the **Tools & Resources** link under Documentation & Tools. Choose **Cisco Product Identification Tool** from the Alphabetical Index drop-down list, or click the **Cisco Product Identification Tool** link under Alerts & RMAs. The CPI tool offers three search options: by product ID or model name; by tree view; or for certain products, by copying and pasting **show** command output. Search results show an illustration of your product with the serial number label location highlighted. Locate the serial number label on your product and record the information before placing a service call.

Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool provides recommended solutions. If your issue is not resolved using the recommended resources, your service request is assigned to a Cisco engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553-2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts

Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, documentation, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com
- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet
- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

  http://www.cisco.com/go/iqmagazine

  or view the digital edition at this URL:

  http://ciscoiq.texterity.com/ciscoiq/sample/
- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

  http://www.cisco.com/ipj
- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:

  http://www.cisco.com/en/US/products/index.html
- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:

  http://www.cisco.com/discuss/networking
- World-class networking training is available from Cisco. You can view current offerings at this URL:

  http://www.cisco.com/en/US/learning/index.html

Implementing BGP on Cisco IOS XR Software

The Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) that allows you to create loop-free interdomain routing between autonomous systems. An autonomous system is a set of routers under a single technical administration. Routers in an autonomous system can use multiple Interior Gateway Protocols (IGP) to exchange routing information inside the autonomous system and an EGP to route packets outside the autonomous system.

This module describes information that is unique to BGP for IP Version 4 (IPv4) and IP Version 6 (IPv6) implementation in Cisco IOS XR Software.

Note For more information about BGP on the Cisco IOS XR software and complete descriptions of the BGP commands listed in this module, you can see the "Related Documents" section of this module. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing BGP on Cisco IOS XR Configuration Module

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.

Contents

Prerequisites for Implementing BGP on Cisco IOS XR Software

To use this command, you must be in a user group associated with a task group that includes the proper task IDs. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.

Information About Implementing BGP on Cisco IOS XR Software

To implement BGP, you need to understand the following concepts:

BGP Functional Overview

BGP uses TCP as its transport protocol. Two BGP routers form a TCP connection between one another (peer routers) and exchange messages to open and confirm the connection parameters.

BGP routers exchange network reachability information. This information is mainly an indication of the full paths (BGP autonomous system numbers) that a route should take to reach the destination network. This information helps construct a graph that shows which autonomous systems are loop free and where routing policies can be applied to enforce restrictions on routing behavior.

Any two routers forming a TCP connection to exchange BGP routing information are called peers or neighbors. BGP peers initially exchange their full BGP routing tables. After this exchange, incremental updates are sent as the routing table changes. BGP keeps a version number of the BGP table, which is the same for all of its BGP peers. The version number changes whenever BGP updates the table due to routing information changes. Keepalive packets are sent to ensure that the connection is alive between the BGP peers and notification packets are sent in response to error or special conditions.

BGP Router Identifier

For BGP sessions between neighbors to be established, BGP must be assigned a router ID. The router ID is sent to BGP peers in the OPEN message when a BGP session is established.

BGP attempts to obtain a router ID in the following ways (in order of preference):

- By means of the address configured using the **bgp router-id** command in router configuration mode.
- By assigning a primary IPv4 address to the interface specified using the **bgp router-id** command in router configuration mode.

Note If the specified interface does not have an IPv4 address, or is not up, BGP will fail to obtain a router ID.

- By using the address specified with the **router-id** command in global configuration mode if the router is booted with the saved **router-id** command and if the ID from this command is available when the last saved loopback configuration is applied.
- By using the primary IPv4 address on the interface specified with the **router-id** command in global configuration mode if the box is booted with the saved **router-id** command in global configuration mode and if the router ID is up by the time all saved loopback configurations are applied.
- By using the highest IPv4 address on a loopback interface in the system if the router is booted with saved loopback address configuration.
- By using the primary IPv4 address of the first loopback address that gets configured if there are not any in the saved configuration.

If none of these methods for obtaining a router ID succeeds, BGP does not have a router ID and cannot establish any peering sessions with BGP neighbors. In such an instance, an error message is entered in the system log, and the **show bgp summary** command displays a router ID of 0.0.0.0.

After BGP has obtained a router ID, it continues to use it even if a better router ID becomes available. This usage avoids unnecessary flapping for all BGP sessions. However, if the router ID currently in use becomes invalid (because the interface goes down or its configuration is changed), BGP selects a new router ID (using the rules described) and all established peering sessions are reset.

We strongly recommend that the **bgp router-id** command is configured to prevent unnecessary changes to the router ID (and consequent flapping of BGP sessions).

BGP Default Limits

Cisco IOS XR BGP imposes maximum limits on the number of neighbors that can be configured on the router and on the maximum number of prefixes that are accepted from a peer for a given address family. This limitation safeguards the router from resource depletion caused by misconfiguration, either locally or on the remote neighbor. The following limits apply to BGP configurations:

- The default maximum number of peers that can be configured is 1024. The default can be changed using the **bgp maximum neighbor** command. The *limit* range is 1 to 1500. Any attempt to configure additional peers beyond the maximum limit or set the maximum limit to a number that is less than the number of peers currently configured will fail.
- To prevent a peer from flooding BGP with advertisements, a limit is placed on the number of prefixes that are accepted from a peer for each supported address family. The default limits can be overridden through configuration of the **maximum-prefix** *limit* command for the peer for the appropriate address family. The following default limits are used if the user does not configure the maximum number of prefixes for the address family:

- 512K (524,288) prefixes for IPv4 unicast.
- 128K (131,072) prefixes for IPv4 multicast.
- 128K (131,072) prefixes for IPv6 unicast.

A cease notification message is sent to the neighbor and the peering with the neighbor is terminated when the number of prefixes received from the peer for a given address family exceeds the maximum limit (either set by default or configured by the user) for that address family.

It is possible that the maximum number of prefixes for a neighbor for a given address family has been configured after the peering with the neighbor has been established and a certain number of prefixes have already been received from the neighbor for that address family. A cease notification message is sent to the neighbor and peering with the neighbor is terminated immediately after the configuration if the configured maximum number of prefixes is fewer than the number of prefixes that have already been received from the neighbor for the address family.

BGP Validation of Local Next-Hop Addresses

When Cisco IOS XR BGP receives a route advertisement from a neighbor, it validates the next-hop address contained in the route by verifying that the next-hop address is not the same as an IP address assigned to an interface on this router (for example, a local address). If the received next-hop address is a local address, the update is dropped. However, if the next-hop address is set to a local address by the configured inbound policy, the update is not dropped, is treated as a valid next-hop address, and is processed normally in Cisco IOS XR BGP. This verification means that the router advertises to its neighbors that it has a route to the prefix, but any traffic received for that prefix is dropped.

This “blackholing” effect is often used to automatically protect against Denial of Service (DOS) attacks on user hosts. An inbound policy is configured that sets the next hop to a local address (for example, the address of a loopback interface) when a route with a particular community is received. When a user finds that a host is under a DOS attack, a BGP advertisement is sent to the address of the attacked host with the special community attached. The advertisement causes the Internet service provider (ISP) router to install a route with a local next hop for that address that drops all traffic destined for it.

BGP Configuration

Cisco IOS XR BGP follows a neighbor-based configuration model that requires that all configurations for a particular neighbor be grouped in one place under the neighbor configuration. Peer groups are not supported for either sharing configuration between neighbors or for sharing update messages. The concept of peer group has been replaced by a set of configuration groups to be used as templates in BGP configuration and automatically generated update groups to share update messages between neighbors. BGP configurations are grouped into four major categories:

- Router Configuration Mode
- Global Address Family Configuration Mode
- Neighbor Configuration Mode
- Neighbor Address Family Configuration Mode

Configuration Modes

The following sections show how to enter each of the configuration modes. From a mode, you can enter the **?** command to display the commands available in that mode.

Router Configuration Mode

The following example shows how to enter router configuration mode:

```
RP/0/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)#
```

Global Address Family Configuration Mode

The following example shows how to enter global address family configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 multicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

Neighbor Configuration Mode

The following example shows how to enter neighbor configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Neighbor Address Family Configuration Mode

The following example shows how to enter neighbor address family configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
```

Neighbor Submode

Cisco IOS XR BGP uses a neighbor submode to make it possible to enter configurations without having to prefix every configuration with the **neighbor** keyword and the neighbor address:

- Cisco IOS XR software has a submode available for neighbors in which it is not necessary for every command to have a "neighbor *x.x.x.x*" prefix.

  In Cisco IOS XR software, the configuration is as follows:

  ```
  Router(config-bgp-af)# neighbor 192.23.1.2
  Router(config-bgp-nbr)# remote-as 2002
  Router(config-bgp-nbr)# address-family ipv4 multicast
  ```

- An address family configuration submode inside the neighbor configuration submode is available for entering address family-specific neighbor configurations. In Cisco IOS XR, the configuration is as follows:

  ```
  Router(config-bgp-af)# neighbor 2002::2
  Router(config-bgp-nbr)# remote-as 2002
  Router(config-bgp-nbr)# address-family ipv6 unicast
  Router(config-bgp-nbr-af)# next-hop-self
  Router(config-bgp-nbr-af)# route-policy one in
  ```

- You must enter neighbor-specific IPv4 or IPv6 commands in neighbor address-family configuration submode. In Cisco IOS XR software, the configuration is as follows:

  ```
  Router(config-bgp)# router bgp 109
  Router(config-bgp)# neighbor 192.168.40.24
  ```

```
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# maximum-prefix 1000
```

Configuration Templates

The **af-group**, **session-group**, and **neighbor-group** configuration commands provide template support for the neighbor configuration in Cisco IOS XR software:

The **af-group** command is used to group address family-specific neighbor commands within an IPv4 or IPv6 address family. Neighbors that have the same address family configuration are able to use the address family group (af-group) name for their address family-specific configuration. A neighbor inherits the configuration from an address family group by way of the **use** command. If a neighbor is configured to use an address family group, the neighbor (by default) inherits the entire configuration from the address family group. However, a neighbor does not inherit all of the configuration from the address family group if items are explicitly configured for the neighbor. The address family group configuration is entered under the BGP router configuration mode. The following example shows how to enter address family group configuration mode.

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group afmcast1 address-family ipv4 multicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)#
```

The **session-group** command allows you to create a session group from which neighbors can inherit address family-independent configuration. A neighbor inherits the configuration from a session group by way of the **use** command. If a neighbor is configured to use a session group, the neighbor (by default) inherits the entire configuration of the session group. A neighbor does not inherit all of the configuration from a session group if a configuration is done directly on that neighbor. The following example shows how to enter session group configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group session1
RP/0/RP0/CPU0:router(config-bgp-sngrp)#
```

The **neighbor-group** command helps you apply the same configuration to one or more neighbors. Neighbor groups can include session groups and address family groups and can comprise the complete configuration for a neighbor. After a neighbor group is configured, a neighbor can inherit the configuration of the group using the **use** command. If a neighbor is configured to use a neighbor group, the neighbor inherits the entire BGP configuration of the neighbor group.

The following example shows how to enter neighbor group configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#
```

The following example shows how to enter neighbor group address family configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)#
```

- However, a neighbor does not inherit all of the configuration from the neighbor group if items are explicitly configured for the neighbor. In addition, some part of the configuration of the neighbor group could be hidden if a session group or address family group was also being used.

Configuration grouping has the following effects in Cisco IOS XR software:

- Commands entered at the session group level define address family-independent commands (the same commands as in the neighbor submode).
- Commands entered at the address family group level define address family-dependent commands for a specified address family (the same commands as in the neighbor-address family configuration submode).
- Commands entered at the neighbor group level define address family-independent commands and address family-dependent commands for each address family (the same as all available **neighbor** commands), and define the **use** command for the address family group and session group commands.

Template Inheritance Rules

In Cisco IOS XR software, BGP neighbors or groups inherit configuration from other configuration groups.

For address family-independent configurations:

- Neighbors can inherit from session groups and neighbor groups.
- Neighbor groups can inherit from session groups and other neighbor groups.
- Session groups can inherit from other session groups.
- If a neighbor uses a session group and a neighbor group, the configurations in the session group are preferred over the global address family configurations in the neighbor group.

For address family-dependent configurations:

- Address family groups can inherit from other address family groups.
- Neighbor groups can inherit from address family groups and other neighbor groups.
- Neighbors can inherit from address family groups and neighbor groups.

Configuration group inheritance rules are numbered in order of precedence as follows:

1. If the item is configured directly on the neighbor, that value is used. In the example that follows, the advertisement interval is configured both on the neighbor group and neighbor configuration and the advertisement interval being used is from the neighbor configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# advertisement-interval 20
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 20 seconds:

```
RP/0/RP0/CPU0:router# show bgp neighbors 10.1.1.1

BGP neighbor is 10.1.1.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 20 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:00:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

2. Otherwise, if the neighbor uses a session group or address family group, the configuration value is obtained from the session group or address family group. If the address family group or session group has a parent and an item is configured on the parent, the parent configuration is used. If the item is not configured on the parent, but is configured on the parent 's parent, the configuration of the parent's parent is used, and so on. In the example that follows, the advertisement interval is configured on a neighbor group and a session group and the advertisement interval value being used is from the session group:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.0.1

BGP neighbor is 192.168.0.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:03:23, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

3. Otherwise, if the neighbor uses a neighbor group and does not use a session group or address family group, the configuration value can be obtained from the neighbor group either directly or through inheritance. In the example that follows, the advertisement interval from the neighbor group is used because it is not configured directly on the neighbor and no session group is used:

```
RP/0/RP0/CPU0:router(config)# router bgp 150
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.1.1

BGP neighbor is 192.168.2.2, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  Inbound path policy configured
  Policy for incoming advertisements is POLICY_1
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:01:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

To illustrate the same rule, the following example shows how to set the advertisement interval to 15 (from the session group). The timers are set to the default (60/180) because the neighbor uses a session group, thus hiding the **timers** command in the neighbor group. The inbound policy is set to POLICY_1 from the neighbor group.

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group ADV
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group TIMER
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# timers 10 30
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.2.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group ADV
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group TIMER
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.2.2

BGP neighbor is 192.168.2.2, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:02:03, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

4. Otherwise, the default value is used. In the example that follows, neighbor 10.0.101.5 has the minimum time between advertisement runs set to 30 seconds (default) because the neighbor is not configured to use the neighbor configuration or the neighbor group configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# neighbor-group adv_15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 10
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.5
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.10
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group adv_15
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 30 seconds:

```
RP/0/RP0/CPU0:router# show bgp neighbors 10.0.101.5

BGP neighbor is 10.0.101.5, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 30 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.2
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%
Connections established 0; dropped 0
```

```
  Last reset 00:00:25, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

The inheritance rules used when groups are inheriting configuration from other groups are the same as the rules given for neighbors inheriting from groups.

Template Inheritance

You can use the following **show** commands described to monitor BGP inheritance information:

- show bgp neighbors, page RC-11
- show bgp af-group, page RC-12
- show bgp session-group, page RC-13
- show bgp neighbor-group, page RC-14

show bgp neighbors

Use the **show bgp neighbors** command to display information about the BGP configuration for neighbors.

- Use the **configuration** keyword to display the effective configuration for the neighbor, including any settings that have been inherited from session groups, neighbor groups, or address family groups used by this neighbor.
- Use the **inheritance** keyword to display the session groups, neighbor groups, and address family groups from which this neighbor is capable of inheriting configuration .

The **show bgp neighbors** command examples that follow are based on the sample configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# next-hop-self
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp0afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# ebgp-multihop 3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 multicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# default-originate
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 200
```

The following example displays sample output from the **show bgp neighbors** command using the **inheritance** keyword. The example shows that the neighbor inherits session parameters from neighbor group GROUP_1, which in turn inherits from session group GROUP_2. The neighbor inherits IPv4 unicast parameters from address family group GROUP_3 and IPv4 multicast parameters from neighbor group GROUP_1:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.0.1 inheritance

  Session:        n:GROUP_1 s:GROUP_2
  IPv4 Unicast:   a:GROUP_3
  IPv4 Multicast: n:GROUP_1
```

The following example displays sample output from the **show bgp neighbors** command using the **configuration** keyword. The example shows from where each item of configuration was inherited, or if it was configured directly on the neighbor (indicated by [ ]). For example, the **ebgp-multihop 3** command was inherited from neighbor group GROUP_1 and the **next-hop-self** command was inherited from the address family group GROUP_3:

```
RP/0/RP0/CPU0:router# show bgp neighbors 192.168.0.1 configuration

neighbor 192.168.0.1
 remote-as 2                   []
 advertisement-interval 15     [n:GROUP_1 s:GROUP_2]
 ebgp-multihop 3               [n:GROUP_1]
 address-family ipv4 unicast   []
  next-hop-self                [a:GROUP_3]
  route-policy POLICY_1    in        [a:GROUP_3]
  weight 200                   []
 address-family ipv4 multicast [n:GROUP_1]
  default-originate            [n:GROUP_1]
```

show bgp af-group

Use the **show bgp af-group** command to display address family groups:

- Use the **configuration** keyword to display the effective configuration for the address family group, including any settings that have been inherited from address family groups used by this address family group.
- Use the **inheritance** keyword to display the address family groups from which this address family group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors, neighbor groups, and address family groups that inherit configuration from this address family group.

The **show bgp af-group** command examples that follow are based on the this sample configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
RP/0/RP0/CPU0:router(config-bgp-afgrp)# default-originate
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# capability orf prefix-list both
```

The following example displays sample output from the **show bgp af-group** command using the **configuration** keyword. This example shows from where each configuration item was inherited. The **default-originate** command was configured directly on this address family group (indicated by [ ]). The **remove-private-as** command was inherited from address family group GROUP_2, which in turn inherited from address family group GROUP_3:

```
RP/0/RP0/CPU0:router# show bgp af-group GROUP_1 configuration

af-group GROUP_1 address-family ipv4 unicast
  capability orf prefix-list both           [a:GROUP_2]
  default-originate                         []
  maximum-prefix 2500 75 warning-only       []
  route-policy POLICY_1 in                  [a:GROUP_2 a:GROUP_3]
  remove-private-AS                         [a:GROUP_2 a:GROUP_3]
  send-community-ebgp                       [a:GROUP_2]
  send-extended-community-ebgp              [a:GROUP_2]
```

The following example displays sample output from the **show bgp af-group** command using the **users** keyword:

```
RP/0/RP0/CPU0:router# show bgp af-group GROUP_2 users

IPv4 Unicast: a:GROUP_1
```

The following example displays sample output from the **show bgp af-group** command using the **inheritance** keyword. This shows that the specified address family group GROUP_1 directly uses the GROUP_2 address family group, which in turn uses the GROUP_3 address family group:

```
RP/0/RP0/CPU0:router# show bgp af-group GROUP_1 inheritance

IPv4 Unicast: a:GROUP_2 a:GROUP_3
```

show bgp session-group

Use the **show bgp session-group** command to display session groups:

- Use the **configuration** keyword to display the effective configuration for the session group, including any settings that have been inherited from session groups used by this session group.
- Use the **inheritance** keyword to display the session groups from which this session group is capable of inheriting configuration.
- Use the **users** keyword to display the session groups, neighbor groups, and neighbors that inherit configuration from this session group.

The examples that follow sample output from the **show bgp session-group** command with the **configuration** keyword in EXEC mode. The examples are based on the following session group configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# update-source Loopback 0
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# ebgp-multihop 2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# dmz-link-bandwidth
```

The following is sample output from the **show bgp session-group** command with the **configuration** keyword in EXEC mode:

```
RP/0/RP0/CPU0:router# show bgp session-group GROUP_1 configuration

session-group GROUP_1
 ebgp-multihop 2         [s:GROUP_2]
 update-source Loopback0 []
 dmz-link-bandwidth      [s:GROUP_2 s:GROUP_3]
```

The following is sample output from the **show bgp session-group** command with the **inheritance** keyword showing that the GROUP_1 session group inherits session parameters from the GROUP_3 and GROUP_2 session groups:

```
RP/0/RP0/CPU0:router# show bgp session-group GROUP_1 inheritance

Session: s:GROUP_2 s:GROUP_3
```

The following is sample output from the **show bgp session-group** command with the **users** keyword showing that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group:

```
RP/0/RP0/CPU0:router# show bgp session-group GROUP_3 users

Session: s:GROUP_1 s:GROUP_2
```

show bgp neighbor-group

Use the **show bgp neighbor-group** command to display neighbor groups:

- Use the **configuration** keyword to display the effective configuration for the neighbor group, including any settings that have been inherited from neighbor groups used by this neighbor group.
- Use the **inheritance** keyword to display the address family groups, session groups, and neighbor groups from which this neighbor group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors and neighbor groups that inherit configuration from this neighbor group.

The examples are based on the following group configuration:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# soft-reconfiguration inbound
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# capability orf prefix-list both
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# timers 30 90
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1982
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# use af-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
```

The following is sample output from the **show bgp neighbor-group** command with the **configuration** keyword. The configuration setting source is shown to the right of each command. In the output shown previously, the remote autonomous system is configured directly on neighbor group GROUP_1, and the send community setting is inherited from neighbor group GROUP_2, which in turn inherits the setting from address family group GROUP_3:

```
RP/0/RP0/CPU0:router# show bgp neighbor-group GROUP_1 configuration

   neighbor-group GROUP_1
    remote-as 1982                   []
    timers 30 90                     [n:GROUP_2 s:GROUP_3]
    address-family ipv4 unicast      []
     capability orf prefix-list both [n:GROUP_2 a:GROUP_2]
     remove-private-AS               [n:GROUP_2 a:GROUP_2 a:GROUP_3]
     send-community-ebgp             [n:GROUP_2 a:GROUP_2]
     send-extended-community-ebgp    [n:GROUP_2 a:GROUP_2]
     soft-reconfiguration inbound    [n:GROUP_2 a:GROUP_2 a:GROUP_3]
     weight 100                      [n:GROUP_2]
```

The following is sample output from the **show bgp neighbor-group** command with the **inheritance** keyword. This output shows that the specified neighbor group GROUP_1 inherits session (address family-independent) configuration parameters from neighbor group GROUP_2. Neighbor group GROUP_2 inherits its session parameters from session group GROUP_3. It also shows that the GROUP_1 neighbor group inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group, which in turn inherits them from the GROUP_2 address family group, which itself inherits them from the GROUP_3 address family group:

```
RP/0/RP0/CPU0:router# show bgp neighbor-group GROUP_1 inheritance

    Session:      n:GROUP-2 s:GROUP_3
    IPv4 Unicast: n:GROUP_2 a:GROUP_2 a:GROUP_3
```

The following is sample output from the **show bgp neighbor-group** command with the **users** keyword. This output shows that the GROUP_1 neighbor group inherits session (address family-independent) configuration parameters from the GROUP_2 neighbor group. The GROUP_1 neighbor group also inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group:

```
RP/0/RP0/CPU0:router# show bgp neighbor-group GROUP_2 users

Session:      n:GROUP_1
IPv4 Unicast: n:GROUP_1
```

No Default Address Family

BGP does not support the concept of a default address family. An address family must be explicitly configured under the BGP router configuration for the address family to be activated in BGP. Similarly, an address family must be explicitly configured under a neighbor for the BGP session to be activated under that address family. It is not required to have any address family configured under the BGP router configuration level for a neighbor to be configured. However, it is a requirement to have an address family configured at the BGP router configuration level for the address family to be configured under a neighbor.

Routing Policy Enforcement

External BGP (eBGP) neighbors must have an inbound and outbound policy configured. If no policy is configured, no routes are accepted from the neighbor, nor are any routes advertised to it. This added security measure ensures that routes cannot accidentally be accepted or advertised in the case of a configuration omission error.

Note This enforcement affects only eBGP neighbors (neighbors in a different autonomous system than this router). For internal BGP (iBGP) neighbors (neighbors in the same autonomous system), all routes are accepted or advertised if there is no policy.

In the following example, for an eBGP neighbor, if all routes should be accepted and advertised with no modifications, a simple pass-all policy is configured:

```
RP/0/RP0/CPU0:router(config)# route-policy pass-all
RP/0/RP0/CPU0:router(config-rpl)# pass
RP/0/RP0/CPU0:router(config-rpl)# end-policy
RP/0/RP0/CPU0:router(config)# commit
```

Use the **route-policy (BGP)** command in the neighbor address-family configuration mode to apply the pass-all policy to a neighbor. The following example shows how to allow all IPv4 unicast routes to be received from neighbor 192.168.40.42 and advertise all IPv4 unicast routes back to it:

```
RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit
```

Use the **show bgp summary** command to display eBGP neighbors that do not have both an inbound and outbound policy for every active address family. In the following example, such eBGP neighbors are indicated in the output with an exclamation (!) mark:

```
RP/0/RP0/CPU0:router# show bgp all all summary

Address Family: IPv4 Unicast
============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 41
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                 41          41          41

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
10.0.101.1        0     1     919     925       41    0    0 15:15:08       10
10.0.101.2        0     2       0       0        0    0    0 00:00:00 Idle


Address Family: IPv4 Multicast
==============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  1           1           1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv4 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
10.0.101.2        0     2       0       0        0    0    0 00:00:00 Idle!


Address Family: IPv6 Unicast
============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 2
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  2           2           2

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
2222::2           0     2     920     918        2    0    0 15:15:11        1
2222::4           0     3       0       0        0    0    0 00:00:00 Idle


Address Family: IPv6 Multicast
==============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  1           1           1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv6 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
2222::2           0     2     920     918        0    0    0 15:15:11        0
2222::4           0     3       0       0        0    0    0 00:00:00 Idle!
```

Table Policy

The table policy feature in BGP allows you to configure traffic index values on routes as they are installed in the global routing table. This feature is enabled using the **table-policy** command and supports the BGP policy accounting feature.

BGP policy accounting uses traffic indices that are set on BGP routes to track various counters. See *Implementing Routing Policy on Cisco IOS XR Software* for details on table policy use. See the *Cisco Express Forwarding Commands on Cisco IOS XR Software* module in the *Cisco IOS XR IP Addresses and Services Command Reference* for details on BGP policy accounting.

Table policy also provides the ability to drop routes from the RIB based on match criteria. This feature can be useful in certain applications and should be used with caution as it can easily create a routing 'black-hole' where BGP advertises routes to neighbors that BGP does not install in its global routing table but in the forwarding table .

Update Groups

The BGP Update Groups feature contains an algorithm that dynamically calculates and optimizes update groups of neighbors that share outbound policies and can share the update messages. The BGP Update Groups feature separates update group replication from peer group configuration, improving convergence time and flexibility of neighbor configuration.

To use this feature, you must understand the following concepts:

- BGP Update Generation and Update Groups, page RC-18
- BGP Update Group, page RC-18

BGP Update Generation and Update Groups

The BGP Update Groups feature separates BGP update generation from neighbor configuration. The BGP Update Groups feature introduces an algorithm that dynamically calculates BGP update group membership based on outbound routing policies. This feature does not require any configuration by the network operator. Update group-based message generation occurs automatically and independently.

BGP Update Group

When a change to the configuration occurs, the router automatically recalculates update group memberships and applies the changes.

For the best optimization of BGP update group generation, we recommend that the network operator keeps outbound routing policy the same for neighbors that have similar outbound policies. This feature contains commands for monitoring BGP update groups. For more information about the commands, see the "Monitoring BGP Update Groups" section on page RC-75.

BGP Best Path Algorithm

BGP routers typically receive multiple paths to the same destination. The BGP best path algorithm determines the best path to install in the IP routing table and to use for forwarding traffic. This section describes the IOS XR implementation of BGP best path algorithm, as specified in Section 9.1 of the Internet Engineering Task Force (IETF) Network Working Group draft-ietf-idr-bgp4-24.txt document.

The BGP best path algorithm implementation is in three parts:

- Part 1—Compares two paths to determine which is better.
- Part 2—Iterates over all paths and determines which order to compare the paths to select the overall best path.
- Part 3—Determines whether the old and new best paths differ enough so that the new best path should be used.

Note The order of comparison determined by Part 2 is important because the comparison operation is not transitive; that is, if three paths, A, B, and C exist, such that when A and B are compared, A is better, and when B and C are compared, B is better, it is not necessarily the case that when A and C are compared, A is better. This nontransitivity arises because the multi exit discriminator (MED) is compared only among paths from the same neighboring autonomous system (AS) and not among all paths.

Comparing Pairs of Paths

The following steps are completed to compare two paths and determine the better path:

1. If either path is invalid (for example, it has the maximum possible MED value, or it has an unreachable nexthop), then the other path is chosen (provided that the path is valid).
2. If the paths have unequal weights, the path with the highest weight is chosen. Note: the weight is entirely local to the router, and can be set with the **weight** command or using a routing policy.
3. If the paths have unequal local preferences, the path with the higher local preference is chosen. Note: If a local preference attribute was received with the path or was set by a routing policy, then that value is used in this comparison. Otherwise, the default local preference value of 100 is used. The default value can be changed using the **bgp default local-preference** command.
4. If one of the paths is a redistributed path, which results from a **redistribute** or **network** command, then it is chosen. Otherwise, if one of the paths is a locally generated aggregate, which results from an **aggregate-address** command, it is chosen.

Note Steps 1 through 4 implement the “Degree of Preference” calculation from Section 9.1.1 of draft-ietf-idr-bgp4-24.txt.

5. If the paths have unequal AS path lengths, the path with the shorter AS path is chosen. This step is skipped if **bgp bestpath as-path ignore** command is configured. Note: when calculating the length of the AS path, confederation segments are ignored, and AS sets count as 1. (See Section 9.1.2.2a of draft-ietf-idr-bgp4-24.txt.)
6. If the paths have different origins, the path with the lower origin is selected. Interior Gateway Protocol (IGP) is considered lower than EGP, which is considered lower than INCOMPLETE. (See Section 9.1.2.2b of draft-ietf-idr-bgp4-24.txt.)
7. If appropriate, the MED of the paths is compared. If they are unequal, the path with the lower MED is chosen.

   A number of configuration options exist that affect whether or not this step is performed. In general, the MED is compared if both paths were received from neighbors in the same AS; otherwise the MED comparison is skipped. However, this behavior is modified by certain configuration options, and there are also some corner cases to consider. (See Section 9.1.2.2c of draft-ietf-idr-bgp4-24.txt.)

If the **bgp bestpath med always** command is configured, then the MED comparison is always performed, regardless of neighbor AS in the paths. Otherwise, MED comparison depends on the AS paths of the two paths being compared, as follows:

a. If a path has no AS path or the AS path starts with an AS_SET, then the path is considered to be internal, and the MED is compared with other internal paths

b. If the AS path starts with an AS_SEQUENCE, then the neighbor AS is the first AS number in the sequence, and the MED is compared with other paths that have the same neighbor AS

c. If the AS path contains only confederation segments or starts with confederation segments followed by an AS_SET, then the MED is not compared with any other path unless the **bgp bestpath med confed** command is configured. In that case, the path is considered internal and the MED is compared with other internal paths.

d. If the AS path starts with confederation segments followed by an AS_SEQUENCE, then the neighbor AS is the first AS number in the AS_SEQUENCE, and the MED is compared with other paths that have the same neighbor AS.

Note: if no MED attribute was received with the path, then the MED is considered to be 0 unless the **bgp bestpath med missing-as-worst** command is configured. In that case, if no MED attribute was received, the MED is considered to be the highest possible value.

8. If one path is received from an external peer and the other is received from an internal (or confederation) peer, the path from the external peer is chosen. (See Section 9.1.2.2d of draft-ietf-idr-bgp4-24.txt.)
9. If the paths have different IGP metrics to their next hops, the path with the lower IGP metric is chosen. (See Section 9.1.2.2e of draft-ietf-idr-bgp4-24.txt.)
10. If all path parameters in steps 1 through 10 are the same, then the router IDs are compared. If the path was received with an originator attribute, then that is used as the router ID to compare; otherwise, the router ID of the neighbor from which the path was received is used. If the paths have different router IDs, the path with the lower router ID is chosen. Note: where the originator is used as the router ID, it is possible to have two paths with the same router ID. It is also possible to have two BGP sessions with the same peer router, and therefore receive two paths with the same router ID. (See Section 9.1.2.2f of draft-ietf-idr-bgp4-24.txt.)
11. If the paths have different cluster lengths, the path with the shorter cluster length is selected. If a path was not received with a cluster list attribute, it is considered to have a cluster length of 0.
12. Finally, the path received from the neighbor with the lower IP address is chosen. Locally generated paths (for example, redistributed paths) are considered to have a neighbor IP address of 0. (See Section 9.1.2.2g of draft-ietf-idr-bgp4-24.txt.)

Order of Comparisons

The second part of the BGP best path algorithm implementation determines the order in which the paths should be compared. The order of comparison is determined as follows:

1. The paths are partitioned into groups such that within each group the MED can be compared among all paths. The same rules as in the "Comparing Pairs of Paths" section on page RC-19 are used to determine whether MED can be compared between any two paths. Normally, this comparison results in one group for each neighbor AS. If the **bgp bestpath med always** command is configured, then there is just one group containing all the paths.

2. The best path in each group is determined. Determining the best path is achieved by iterating through all paths in the group and keeping track of the best one seen so far. Each path is compared with the best-so-far, and if it is better, it becomes the new best-so-far and is compared with the next path in the group.
3. A set of paths is formed containing the best path selected from each group in step 2. The overall best path is selected from this set of paths, by iterating through them as in step 2.

Best Path Change Suppression

The third part of the implementation is to determine whether the best path change can be suppressed or not—whether the new best path should be used, or continue using the existing best path. The existing best path can continue to be used if the new one is identical to the point at which the best path selection algorithm becomes arbitrary (if the router-id is the same). Continuing to use the existing best path can avoid churn in the network.

Note This suppression behavior does not comply with the IETF Networking Working Group draft-ietf-idr-bgp4-24.txt document, but is specified in the IETF Networking Working Group draft-ietf-idr-avoid-transition-00.txt document.

The suppression behavior can be turned off by configuring the **bgp bestpath compare-routerid** command. If this command is configured, the new best path is always preferred to the existing one.

Otherwise, the following steps are used to determine whether the best path change can be suppressed:

1. If the existing best path is no longer valid, the change cannot be suppressed.
2. If either the existing or new best paths were received from internal (or confederation) peers or were locally generated (for example, by redistribution), then the change cannot be suppressed. That is, suppression is possible only if both paths were received from external peers.
3. If the paths were received from the same peer (the paths would have the same router-id), the change cannot be suppressed. The router ID is calculated using rules in the "Comparing Pairs of Paths" section on page RC-19.
4. If the paths have different weights, local preferences, origins, or IGP metrics to their next hops, then the change cannot be suppressed. Note that all of these values are calculated using the rules in the "Comparing Pairs of Paths" section on page RC-19.
5. If the paths have different-length AS paths and the **bgp bestpath as-path ignore** command is not configured, then the change cannot be suppressed. Again, the AS path length is calculated using the rules in the "Comparing Pairs of Paths" section on page RC-19.
6. If the MED of the paths can be compared and the MEDs are different, then the change cannot be suppressed. The decision as to whether the MEDs can be compared is exactly the same as the rules in the "Comparing Pairs of Paths" section on page RC-19, as is the calculation of the MED value.
7. If all path parameters in steps 1 through 6 do not apply, the change can be suppressed.

Multiprotocol BGP

Multiprotocol BGP is an enhanced BGP that carries routing information for multiple network layer protocols and IP multicast routes. BGP carries two sets of routes, one set for unicast routing and one set for multicast routing. The routes associated with multicast routing are used by the Protocol Independent Multicast (PIM) feature to build data distribution trees.

Multiprotocol BGP is useful when you want a link dedicated to multicast traffic, perhaps to limit which resources are used for which traffic. Multiprotocol BGP allows you to have a unicast routing topology different from a multicast routing topology providing more control over your network and resources.

In BGP, the only way to perform interdomain multicast routing was to use the BGP infrastructure that was in place for unicast routing. Perhaps you want all multicast traffic exchanged at one network access point (NAP). If those routers were not multicast capable, or there were differing policies for which you wanted multicast traffic to flow, multicast routing could not be supported without multiprotocol BGP.

Note It is possible to configure BGP peers that exchange both unicast and multicast network layer reachability information (NLRI), but you cannot connect multiprotocol BGP clouds with a BGP cloud. That is, you cannot redistribute multiprotocol BGP routes into BGP.

Figure 1 illustrates simple unicast and multicast topologies that are incongruent, and therefore are not possible without multiprotocol BGP.

Autonomous systems 100, 200, and 300 are each connected to two NAPs that are FDDI rings. One is used for unicast peering (and therefore the exchange of unicast traffic). The Multicast Friendly Interconnect (MFI) ring is used for multicast peering (and therefore the exchange of multicast traffic). Each router is unicast and multicast capable.

Figure 1 Incongruent Unicast and Multicast Routes

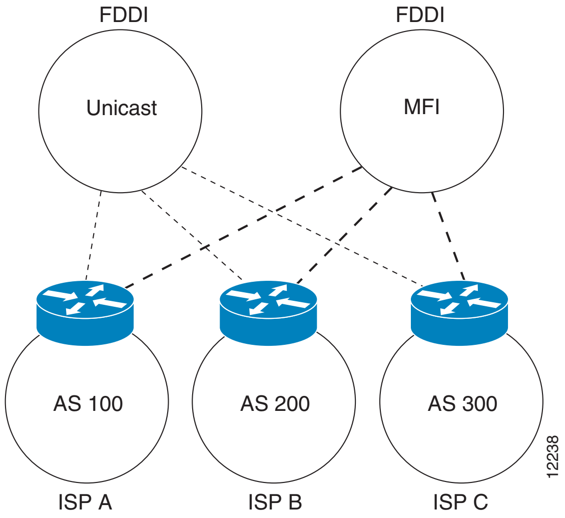


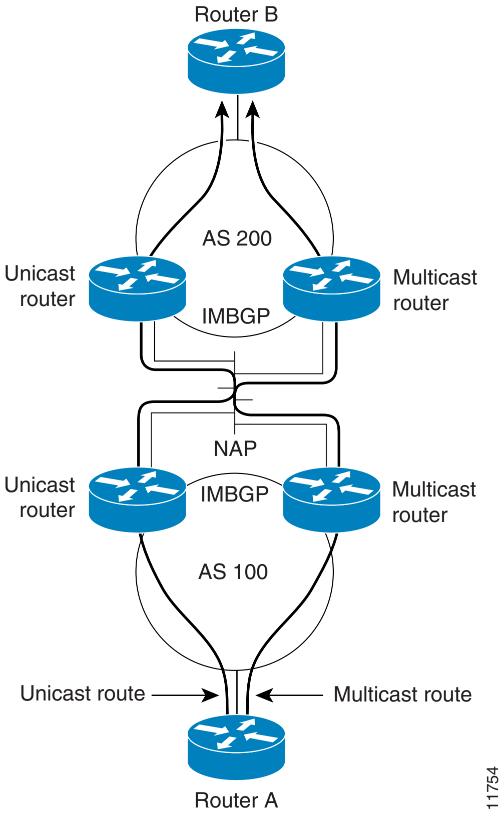
Figure 2 is a topology of unicast-only routers and multicast-only routers. The two routers on the left are unicast-only routers (that is, they do not support or are not configured to perform multicast routing). The two routers on the right are multicast-only routers. Routers A and B support both unicast and multicast routing. The unicast-only and multicast-only routers are connected to a single NAP.

In Figure 2, only unicast traffic can travel from Router A to the unicast routers to Router B and back. Multicast traffic could not flow on that path, so another routing table is required. Multicast traffic uses the path from Router A to the multicast routers to Router B and back.

Figure 2 illustrates a multiprotocol BGP environment with a separate unicast route and multicast route from Router A to Router B. Multiprotocol BGP allows these routes to be incongruent. Both of the autonomous systems must be configured for internal multiprotocol BGP (IMBGP) in the figure.

A multicast routing protocol, such as PIM, uses the multicast BGP database to perform Reverse Path Forwarding (RPF) lookups for multicast-capable sources. Thus, packets can be sent and accepted on the multicast topology but not on the unicast topology.

Figure 2 Multicast BGP Environment



Route Dampening

Route dampening is a BGP feature that minimizes the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on.

For example, consider a network with three BGP autonomous systems: autonomous system 1, autonomous system 2, and autonomous system 3. Suppose the route to network A in autonomous system 1 flaps (it becomes unavailable). Under circumstances without route dampening, the eBGP neighbor of autonomous system 1 to autonomous system 2 sends a withdraw message to autonomous system 2. The border router in autonomous system 2, in turn, propagates the withdrawal message to autonomous system 3. When the route to network A reappears, autonomous system 1 sends an advertisement message to autonomous system 2, which sends it to autonomous system 3. If the route to network A repeatedly becomes unavailable, then available, many withdrawal and advertisement messages are sent. Route flapping is a problem in an internetwork connected to the Internet because a route flap in the Internet backbone usually involves many routes.

Note No penalty is applied to a BGP peer reset when route dampening is enabled. Although the reset withdraws the route, no penalty is applied in this instance, even if route flap dampening is enabled.

Minimizing Flapping

The route dampening feature minimizes the flapping problem as follows. Suppose again that the route to network A flaps. The router in autonomous system 2 (in which route dampening is enabled) assigns network A a penalty of 1000 and moves it to history state. The router in autonomous system 2 continues to advertise the status of the route to neighbors. The penalties are cumulative. When the route flaps so often that the penalty exceeds a configurable suppression limit, the router stops advertising the route to network A, regardless of how many times it flaps. Thus, the route is dampened.

The penalty placed on network A is decayed until the reuse limit is reached, upon which the route is once again advertised. At half of the reuse limit, the dampening information for the route to network A is removed.

BGP Routing Domain Confederation

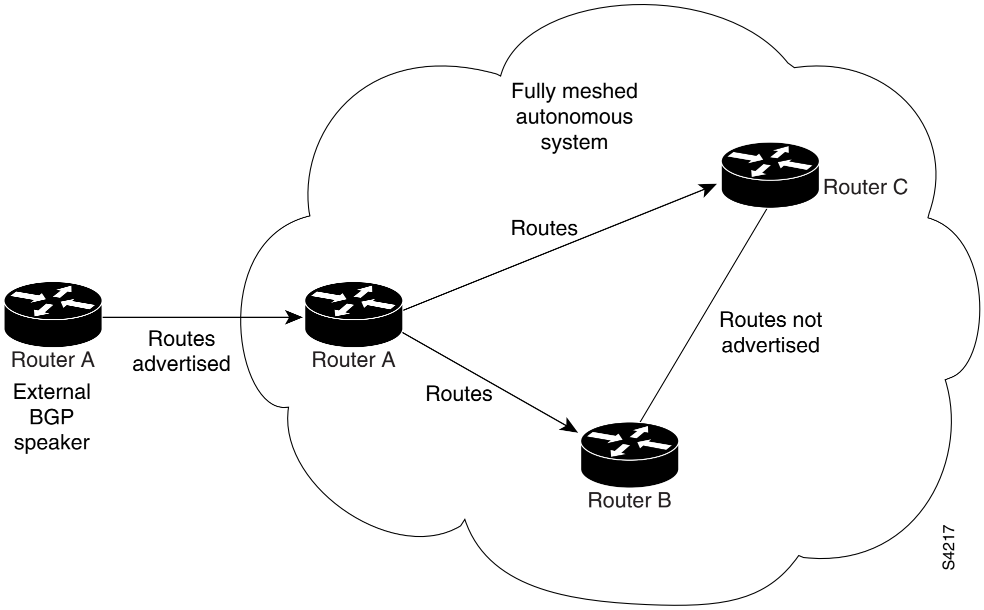
One way to reduce the iBGP mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Although the peers in different autonomous systems have eBGP sessions, they exchange routing information as if they were iBGP peers. Specifically, the next hop, MED, and local preference information is preserved. This feature allows the you to retain a single IGP for all of the autonomous systems.

BGP Route Reflectors

BGP requires that all iBGP speakers be fully meshed. However, this requirement does not scale well when there are many iBGP speakers. Instead of configuring a confederation, another way to reduce the iBGP mesh is to configure a *route reflector*.

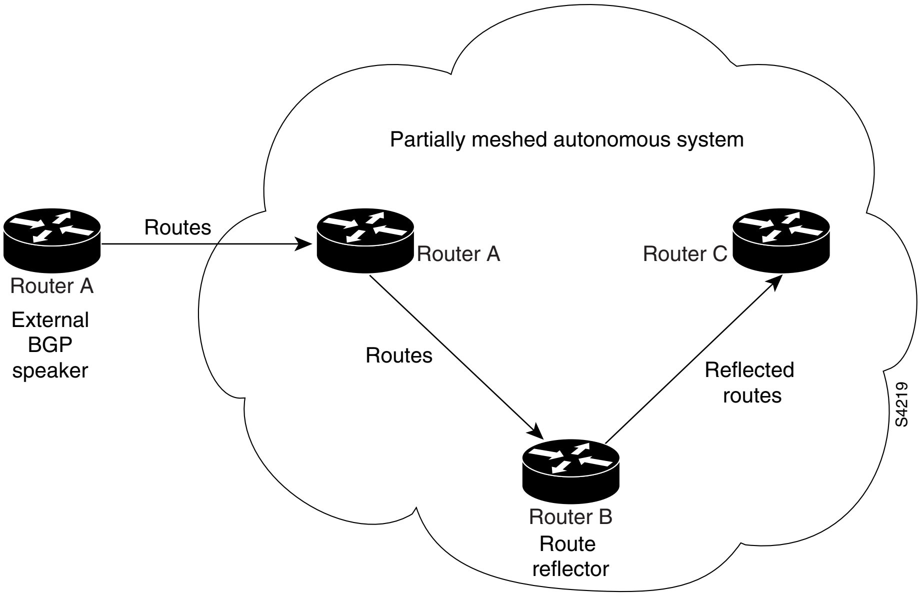
Figure 3 illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise the iBGP learned route to other iBGP speakers because the routers do not pass on routes learned from internal neighbors to other internal neighbors, thus preventing a routing information loop.

Figure 3 Three Fully Meshed iBGP Speakers



With route reflectors, all iBGP speakers need not be fully meshed because there is a method to pass learned routes to neighbors. In this model, an iBGP peer is configured to be a route reflector responsible for passing iBGP learned routes to a set of iBGP neighbors. In Figure 4, Router B is configured as a route reflector. When the route reflector receives routes advertised from Router A, it advertises them to Router C, and vice versa. This scheme eliminates the need for the iBGP session between routers A and C.

Figure 4 Simple BGP Model with a Route Reflector



The internal peers of the route reflector are divided into two groups: client peers and all other routers in the autonomous system (nonclient peers). A route reflector reflects routes between these two groups. The route reflector and its client peers form a *cluster*. The nonclient peers must be fully meshed with each other, but the client peers need not be fully meshed. The clients in the cluster do not communicate with iBGP speakers outside their cluster.

Figure 5 More Complex BGP Route Reflector Model

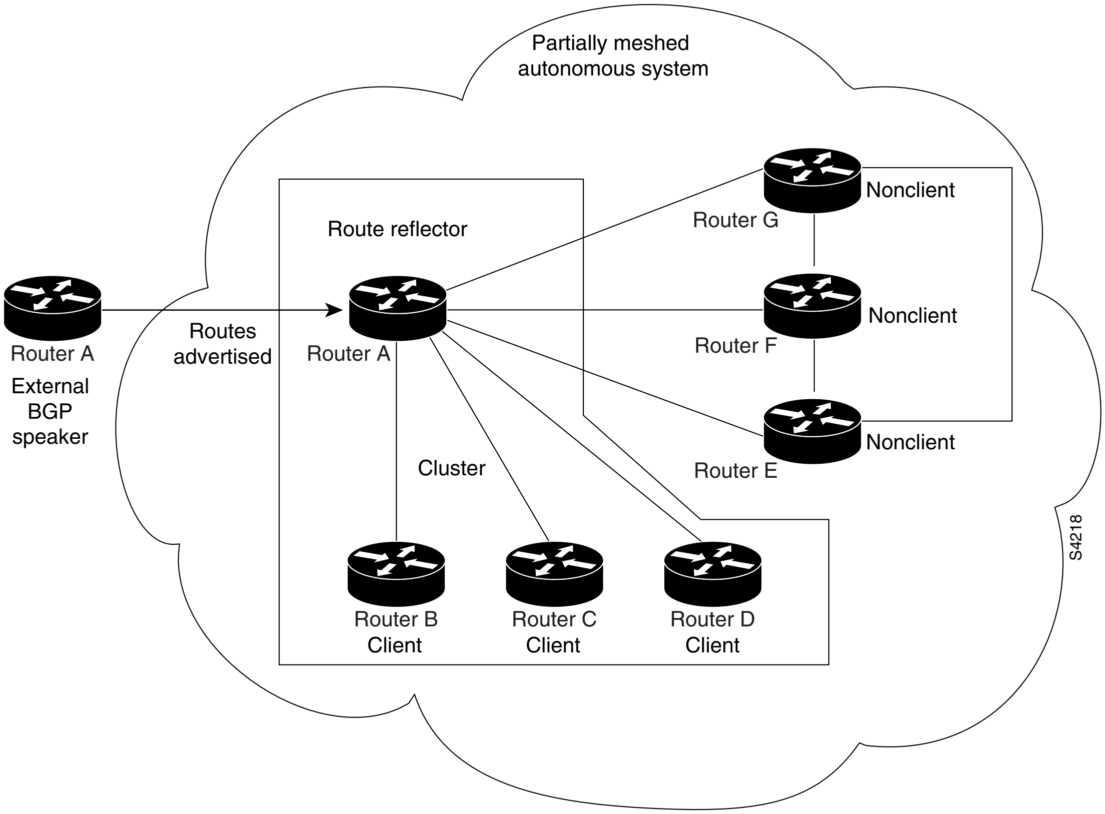


Figure 5 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

When the route reflector receives an advertised route, depending on the neighbor, it takes the following actions:

- A route from an external BGP speaker is advertised to all clients and nonclient peers.
- A route from a nonclient peer is advertised to all clients.
- A route from a client is advertised to all clients and nonclient peers. Hence, the clients need not be fully meshed.

Along with route reflector-aware BGP speakers, it is possible to have BGP speakers that do not understand the concept of route reflectors. They can be members of either client or nonclient groups, allowing an easy and gradual migration from the old BGP model to the route reflector model. Initially, you could create a single cluster with a route reflector and a few clients. All other iBGP speakers could be nonclient peers to the route reflector and then more clusters could be created gradually.

An autonomous system can have multiple route reflectors. A route reflector treats other route reflectors just like other iBGP speakers. A route reflector can be configured to have other route reflectors in a client group or nonclient group. In a simple configuration, the backbone could be divided into many clusters. Each route reflector would be configured with other route reflectors as nonclient peers (thus, all route reflectors are fully meshed). The clients are configured to maintain iBGP sessions with only the route reflector in their cluster.

Usually, a cluster of clients has a single route reflector. In that case, the cluster is identified by the router ID of the route reflector. To increase redundancy and avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All route reflectors serving a cluster should be fully meshed and all of them should have identical sets of client and nonclient peers.

By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, the route reflector need not reflect routes to clients.

As the iBGP learned routes are reflected, routing information may loop. The route reflector model has the following mechanisms to avoid routing loops:

- Originator ID is an optional, nontransitive BGP attribute. It is a 4-byte attributed created by a route reflector. The attribute carries the router ID of the originator of the route in the local autonomous system. Therefore, if a misconfiguration causes routing information to come back to the originator, the information is ignored.
- Cluster-list is an optional, nontransitive BGP attribute. It is a sequence of cluster IDs that the route has passed. When a route reflector reflects a route from its clients to nonclient peers, and vice versa, it appends the local cluster ID to the cluster-list. If the cluster-list is empty, a new cluster-list is created. Using this attribute, a route reflector can identify if routing information is looped back to the same cluster due to misconfiguration. If the local cluster ID is found in the cluster-list, the advertisement is ignored.

Default Address Family for show Commands

Most of the **show** commands require the address family (afi) and subsequent address family (safi) to be specified as arguments. The Cisco IOS XR software parser provides the ability to set the afi and safi so it is not necessary to specify them while executing a **show** command. The parser commands are:

- **set default-afi** {**ipv4** | **ipv6** | **all**}
- **set default-safi** {**unicast** | **multicast** | **all**}

The parser automatically sets the default afi value to **ipv4** and default safi value to **unicast**. It is necessary to use only the parser commands to change the default afi value from **ipv4** or default safi value from **unicast**. Any **afi** or **safi** keyword specified in a **show** command overrides the values set using the parser commands. Use the following command to check the currently set value of the afi and safi:

- **show default-afi-safi**

How to Implement BGP on Cisco IOS XR Software

This section contains instructions for the following tasks:

- Enabling BGP Routing, page RC-28 (required)
- Configuring a Routing Domain Confederation for BGP, page RC-31 (optional)
- Resetting eBGP Session Immediately Upon Link Failure, page RC-33 (optional)
- Logging Neighbor Changes, page RC-34 (optional)
- Adjusting BGP Timers, page RC-34 (optional)
- Changing the BGP Default Local Preference Value, page RC-35 (optional)
- Configuring the MED Metric for BGP, page RC-36 (optional)
- Configuring BGP Weights, page RC-38 (optional)

- Tuning the BGP Best Path Calculation, page RC-39 (optional)
- Indicating BGP Backdoor Routes, page RC-41 (optional)
- Configuring Aggregate Addresses, page RC-43 (optional)
- Redistributing iBGP Routes into IGP, page RC-44 (optional)
- Redistributing Prefixes into Multiprotocol BGP, page RC-46 (optional)
- Configuring BGP Route Dampening, page RC-48 (optional)
- Applying Policy When Updating the Routing Table, page RC-52 (optional)
- Setting BGP Administrative Distance, page RC-53 (optional)
- Configuring a BGP Neighbor Group, page RC-55 (optional)
- Configuring a BGP Neighbor, page RC-58 (required)
- Configuring a Route Reflector for BGP, page RC-60 (optional)
- Configuring BGP Route Filtering by Route Policy, page RC-62 (optional)
- Disabling Next Hop Processing on BGP Updates, page RC-64 (optional)
- Configuring BGP Community and Extended-Community Filtering, page RC-65 (optional)
- Configuring Software to Store Updates from a Neighbor, page RC-67 (optional)
- Disabling a BGP Neighbor, page RC-69 (optional)
- Resetting Neighbors Using BGP Dynamic Inbound Soft Reset, page RC-71 (optional)
- Resetting Neighbors Using BGP Outbound Soft Reset, page RC-71 (optional)
- Resetting Neighbors Using BGP Hard Reset, page RC-72 (optional)
- Clearing Caches, Tables and Databases, page RC-73 (optional)
- Displaying System and Network Statistics, page RC-73 (optional)
- Monitoring BGP Update Groups, page RC-75 (optional)

Enabling BGP Routing

Perform this task to enable BGP routing and establish a BGP routing process. Configuring BGP neighbors is included as part of enabling BGP routing.

Note At least one neighbor and at least one address family must be configured to enable BGP routing. At least one neighbor with both a remote AS and an address family must be configured globally using the **address family** and **remote as** commands.

Prerequisites

BGP must be able to obtain a router identifier (for example, a configured loopback address). At least, one address family must be configured in the BGP router configuration and the same address family must also be configured under the neighbor.

Restrictions

If the neighbor is configured as an external BGP (eBGP) peer, you must configure an inbound and outbound route policy on the neighbor using the **route-policy** command.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **end-policy**
4. **end**
or
commit
5. **configure**
6. **router bgp** *autonomous-system-number*
7. **bgp router-id** {*ip-address* | *interface-type interface-instance*}
8. **neighbor** *ip-address*
9. **remote-as** *autonomous-system-number*
10. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
11. **route-policy** *route-policy-name* {**in** | **out**}
12. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`route-policy`** *`name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234`
	`RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`RP/0/RP0/CPU0:router(config-rpl)# apply check-communities`
	`RP/0/RP0/CPU0:router(config-rpl)# else`
	`RP/0/RP0/CPU0:router(config-rpl)# pass`
	`RP/0/RP0/CPU0:router(config-rpl)# endif`
		(Optional) Defines a route policy named drop-as-1234 and enters route policy configuration mode.

	Command or Action	Purpose

Step 3	**end-policy**
	Example:
	`RP/0/RP0/CPU0:router(config-rpl)# end-policy`
		(Optional) Ends the definition of a route policy and exits route policy configuration mode.

Step 4	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 6	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 7	**bgp router-id** {*ip-address* | *interface-type interface-instance*}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24`
		Configures the local router with a router id of 192.168.70.24.

Step 8	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

	Command or Action	Purpose
Step 9	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 10	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

Step 11	**route-policy** *route-policy-name* {**in** | **out**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`
		(Optional) Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

Step 12	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Routing Domain Confederation for BGP

Perform this task to configure the routing domain confederation for BGP. This includes specifying a confederation identifier and autonomous systems that belong to the confederation.

Configuring a routing domain confederation reduces the internal BGP (iBGP) mesh by dividing an autonomous system into multiple autonomous systems and grouping them into a single confederation. Each autonomous system is fully meshed within itself and has a few connections to another autonomous system in the same confederation. The confederation maintains the next hop and local preference information, and that allows you to retain a single Interior Gateway Protocol (IGP) for all autonomous systems. To the outside world, the confederation looks like a single autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp confederation identifier** *autonomous-system-number*
4. **bgp confederation peers** *autonomous-system-number*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`bgp confederation identifier`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5`
		Specifies a BGP confederation identifier of 5.

	Command or Action	Purpose

Step 4	**`bgp confederation peers`**
	`autonomous-system-number`
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1091`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1092`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1093`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1094`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1095`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp`
	`confederation peers 1096`
		Specifies that the BGP autonomous systems 1091, 1092, 1093, 1094, 1095, and 1096 belong to BGP confederation identifier 5.

Step 5	**`end`**
	or
	`commit`
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before`
		`exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Resetting eBGP Session Immediately Upon Link Failure

Immediately resetting BGP sessions of any directly adjacent external peers if the link used to reach them goes down is enabled by default. Use the **bgp fast-external-fallover disable** command to disable automatic resetting. The **bgp fast-external-fallover disable** command can also be used to turn the automatic reset back on.

Logging Neighbor Changes

Logging neighbor changes is enabled by default. Use the **log neighbor changes disable** command to turn off logging. The **log neighbor changes disable** command can also be used to turn logging back on if it has been disabled.

Adjusting BGP Timers

Perform this task to set the timers for BGP neighbors.

BGP uses certain timers to control periodic activities, such as the sending of keepalive messages and the interval after which a neighbor is assumed to be down if no messages are received from the neighbor during the interval. The values set using the **timers bgp** command can be overridden on particular neighbors using the **timers** command in the neighbor configuration mode.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **timers bgp** *keepalive hold-time*
4. **neighbor** *ip-address*
5. **timers** *keepalive hold-time*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`timers bgp`** *`keepalive hold-time`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# timers bgp 30 90`
		Sets a default keepalive time of 30 seconds and a default hold time of 90 seconds for all neighbors.

	Command or Action	Purpose
Step 4	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.
Step 5	**timers** *keepalive hold-time* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# timers 60 220`	(Optional) Sets the keepalive timer to 60 seconds and the hold-time timer to 220 seconds for BGP neighbor 172.168.40.24.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Changing the BGP Default Local Preference Value

Perform this task to set the default local preference value for BGP paths.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp default local-preference** *value*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`bgp default local-preference`** *`value`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp default local-preference 200`
		Sets the default local preference value from the default of 100 to 200, making it a more preferable path.

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the MED Metric for BGP

Perform this task to set the multi exit discriminator (MED) to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **default-metric** *value*

4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`default-metric`** *`value`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# default metric 10`
		Sets the default metric to 10, which is used to set the MED to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP Weights

Perform this task to assign a weight to routes received from a neighbor. A weight is a number that you can assign to a path so that you can control the best path selection process. If you have particular neighbors that you want to prefer for most of your traffic, you can use the **weight** command to assign a higher weight to all routes learned from that neighbor.

Restrictions

The **clear bgp** command must be used for the newly configured weight to take effect.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
6. **weight** *weight-value*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

	Command or Action	Purpose

Step 5	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

Step 6	**weight** *weight-value* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 41150`	Assigns a weight of 41150 to all IPv4 unicast routes learned through 172.168.40.24.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Tuning the BGP Best Path Calculation

Perform this task to change the default BGP best path calculation behavior.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp bestpath med missing-as-worst**
4. **bgp bestpath med always**
5. **bgp bestpath med confed**
6. **bgp bestpath as-path ignore**
7. **bgp bestpath compare-routerid**

8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 3	**bgp bestpath med missing-as-worst** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med missing-as-worst`	Directs the BGP software to consider a missing MED attribute in a path as having a value of infinity, making this path the least desirable path.
Step 4	**bgp bestpath med always** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med always`	Configures the BGP speaker in autonomous system 120 to compare MEDs among alternative paths, regardless of the autonomous system from which the paths are received.
Step 5	**bgp bestpath med confed** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath med confed`	Enables BGP software to compare MED values for paths learned from confederation peers.
Step 6	**bgp bestpath as-path ignore** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath as-path ignore`	Configures the BGP software to ignore the autonomous system length when performing best path selection.

	Command or Action	Purpose
Step 7	**bgp bestpath compare-routerid** **Example:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath compare-routerid`	Configure the BGP speaker in autonomous system 120 to compare the router IDs of similar paths.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Indicating BGP Backdoor Routes

Perform this task to set the administrative distance on an external Border Gateway Protocol (eBGP) route to that of a locally sourced BGP route, causing it to be less preferred than an Interior Gateway Protocol (IGP) route.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **network** {*ip-address /prefix-length* | *ip-address mask*} **backdoor**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family`** `{`**`ipv4 unicast`** `|` **`ipv4 multicast`** `|` **`ipv6 unicast`** `|` **`ipv6 multicast`**`}`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

Step 4	**`network`** `{`*`ip-address /prefix-length`* `|` *`ip-address mask`*`}` **`backdoor`**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# network 172.20.0.0/16`
		Configures the local router to originate and advertise the IPv4 unicast network 172.20.0.0/16.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Aggregate Addresses

Perform this task to create aggregate entries in a BGP routing table.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 4	**aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set`	Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized. • The **as-set** keyword generates autonomous system set path information and community information from contributing paths. • The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths. • The **summary-only** keyword filters all more specific routes from updates. • The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Redistributing iBGP Routes into IGP

Perform this task to redistribute iBGP routes into an Interior Gateway Protocol (IGP), such as Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF).

Note Use of the **bgp redistribute-internal** command requires the **clear route *** command to be issued to reinstall all BGP routes into the IP routing table.

Caution Redistributing iBGP routes into IGPs may cause routing loops to form within an autonomous system. Use this command with caution.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp redistribute-internal**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`bgp redistribute-internal`**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp redistribute-internal`
		Allows the redistribution of iBGP routes into an IGP, such as IS-IS or OSPF.

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Redistributing Prefixes into Multiprotocol BGP

Perform this task to redistribute prefixes from another protocol into multiprotocol BGP.

Redistribution is the process of injecting prefixes from one routing protocol into another routing protocol. This task shows how to inject prefixes from another routing protocol into multiprotocol BGP. Specifically, prefixes that are redistributed into multiprotocol BGP using the **redistribute** command are injected into the unicast database, the multicast database, or both.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute isis *process-id* [**level** {1 | 1-inter-area | 2}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospfv3 *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute static [**metric** *metric-value*] [**route-policy** *route-policy-name*]
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 4	**redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute isis** *process-id* [**level** {1 | 1-inter-area | 2}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute ospfv3** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# redistribute ospf 110`	Causes IPv4 unicast OSPF routes from OSPF instance 110 to be redistributed into BGP.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP Route Dampening

Perform this task to configure and monitor BGP route dampening.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]
5. **end**
   or
   commit
6. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics**
7. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics regexp** *regular-expression*
8. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics route-policy** *route-policy-name*
9. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics** {*ip-address* [{*mask* | */prefix-length*}
10. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics** {*ip-address* [{*mask* | */prefix-length*} [**longer-prefixes**]]
11. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics**
12. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics regexp** *regular-expression*
13. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics route-policy** *route-policy-name*
14. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics** *network***/***mask-length*
15. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics** *ip-address*
16. **show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **dampened-paths**
17. **clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **dampening** [*ip-address***/***mask-length*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

Step 4	**bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120`	Configures BGP dampening for the IPv4 address family. The *half-life* argument is set to 30, the *reuse* argument is set to 1500, the *suppress* argument is set to 10000, and the *max-suppress-time* argument is set to 120.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 6	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics** **Example:** `RP/0/RP0/CPU0:router# show bgp flap statistics`	Displays BGP flap statistics for all paths.
Step 7	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics regexp** *regular-expression* **Example:** `RP/0/RP0/CPU0:router# show bgp flap-statistics regexp _1$`	Displays BGP flap statistics for all paths that match the regular expression _1$.
Step 8	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics route-policy** *route-policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# show bgp flap-statistics route-policy policy_A`	Displays BGP flap statistics for route policy policy_A.

Step 9	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics** {*ip-address* [{*mask* | */prefix-length*}
	Example:
	`RP/0/RP0/CPU0:router# show bgp flap-statistics 172.20.1.1`
		Displays BGP flap statistics for neighbor 172.20.1.1.

Step 10	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **flap-statistics** {*ip-address* [{*mask* | */prefix-length*} [**longer-prefixes**] **Example:** `RP/0/RP0/CPU0:router# show bgp flap-statistics 172.20.1.1 longer-prefixes`	Displays BGP flap statistics for more specific entries for neighbor 172.20.1.1.
Step 11	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics** **Example:** `RP/0/RP0/CPU0:router# clear bgp all all flap-statistics`	Clears BGP flap statistics for all routes.

	Command or Action	Purpose
Step 12	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics regexp** *regular-expression* **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics _1$`	Clears BGP flap statistics for all paths that match the regular expression _1$.
Step 13	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics route-policy** *route-policy-nane* **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics route-policy policy_A`	Clears BGP flap statistics for route policy policy_A.
Step 14	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics** *network***/***mask-length* **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 192.168.40.0/24`	Clears BGP flap statistics for network 192.168.40.0/24.
Step 15	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **flap-statistics** *ip-address* **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 172.20.1.1`	Clears BGP flap statistics for routes received from this neighbor 172.20.1.1.
Step 16	**show bgp** [**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}] **dampened-paths** **Example:** `RP/0/RP0/CPU0:router# show bgp dampened paths`	Displays the dampened routes, including the time remaining before they are unsuppressed.
Step 17	**clear bgp** {**ipv4** {**unicast** | **multicast** | **all**} | **ipv6** {**unicast** | **all**} | **all** {**unicast** | **multicast** | **all**}} **dampening** [*ip-address***/***mask-length*] **Example:** `RP/0/RP0/CPU0:router# clear bgp dampening`	Clears route dampening information and unsuppresses the suppressed routes.

Applying Policy When Updating the Routing Table

Perform this task to apply a routing policy to routes being installed into the routing table.

Prerequisites

See the *Implementing Routing Policy on Cisco IOS XR Software* module of the *Cisco IOS XR Routing Configuration Guide* for a list of the supported attributes and operations that are valid for table policy filtering.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **table-policy** *policy-name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 4	**table-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)# table-policy tbl-plcy-A`	Applies the tbl-plcy-A policy to IPv4 unicast routes being installed into the routing table.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Setting BGP Administrative Distance

Perform this task to specify the use of administrative distances that can be used to prefer one class of route over another.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
4. **distance bgp** *external-distance internal-distance local-distance*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family`** `{`**`ipv4 unicast`** `|` **`ipv4 multicast`** `|` **`ipv6 unicast`** `|` **`ipv6 multicast`**`}`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Enters global address family configuration mode for the IPv4 address family.

Step 4	**`distance bgp`** *`external-distance internal-distance local-distance`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# distance bgp 20 20 200`
		Sets the external, internal, and local administrative distances to prefer one class of routes over another. The higher the value, the lower the trust rating. The *external-distance* argument is set to 20, the *internal-distance* argument is set to 20, and the *local-distance* argument is set to 200.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a BGP Neighbor Group

Perform this task to configure BGP neighbor groups and apply the neighbor group configuration to a neighbor.

After a neighbor group is configured, each neighbor can inherit the configuration through the **use** command. If a neighbor is configured to use a neighbor group, the neighbor (by default) inherits the entire configuration of the neighbor group, which includes the address family-independent and address family-dependent configurations. The inherited configuration can be overridden if you directly configure commands for the neighbor or configure session groups or address family groups through the **use** command.

From neighbor group configuration mode, you can configure address family-independent parameters for the neighbor group. Use the **address-family** command when in the neighbor group configuration mode.

After specifying the neighbor group name using the **neighbor group** command, you can assign options to the neighbor group.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor-group** *name*
4. **remote-as** *autonomous-system-number*
5. **advertisement-interval** *seconds*
6. **description** *text*
7. **dmz-link-bandwidth**
8. **ebgp-multihop** [*ttl-value*]
9. **local-as** *autonomous-system-number*
10. **password** {**clear** | **encrypted**} *password*
11. **password-disable**
12. **receive-buffer-size** *socket-size* [*bgp-size*]
13. **send-buffer-size** *socket-size* [*bgp-size*]
14. **timers** *keepalive hold-time*
15. **ttl-security**
16. **update-source** *interface-type interface-number*
17. **exit**
18. **neighbor** *ip-address*
19. **use neighbor-group** *group-name*
20. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor-group`** *`name`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)#`
	`neighbor-group nbr-grp-A`
		Places the router in neighbor group configuration mode.

Step 4	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 5	**`advertisement-interval`** *`seconds`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`advertisement-interval 10`
		(Optional) Sets the minimum time between sending BGP routing updates to 10 seconds.

Step 6	**`description`** *`text`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`description Neighbor on BGP 120`
		(Optional) Configures the description “Neighbor on BGP 120” for neighbor group nbr-grp-A.

Step 7	**`dmz-link-bandwidth`**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`dmz-link-bandwidth`
		(Optional) Advertises the bandwidth of links on router bgp 120.

Step 8	**`ebgp-multihop`** `[`*`ttl-value`*`]`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`ebgp-multihop`
		(Optional) Allows a BGP connection to neighbor group nbr-grp-A.

Step 9	**`local-as`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#`
	`local-as 30`
		(Optional) Specifies that BGP use autonomous system 30 for the purpose of peering with neighbor group nbr-grp-A.

	Command or Action	Purpose
Step 10	**password** {**clear** | **encrypted**} *password* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# password clear pswd123`	(Optional) Configures neighbor group nbr-grp-A to use MD5 authentication with the password pswd123.
Step 11	**password-disable** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# password-disable`	(Optional) Overrides any inherited password configuration from the neighbor group.
Step 12	**receive-buffer-size** *socket-size* [*bgp-size*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# receive-buffer-size 45215 5156`	(Optional) Sets the receive buffer sizes for neighbor group nbr-grp-A to 45215 bytes for the socket buffer and 5156 bytes for the BGP buffer.
Step 13	**send-buffer-size** *socket-size* [*bgp-size*] **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# send-buffer-size 8741 8741`	(Optional) Sets the send buffer sizes for neighbor group nbr-grp-A to 8741 bytes for the socket buffer and 8741 bytes for the BGP buffer.
Step 14	**timers** *keepalive hold-time* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# timers 60 220`	(Optional) Sets the keepalive timer to 60 seconds and the hold-time timer to 220 seconds for the BGP neighbor group nbr-grp-A.
Step 15	**ttl-security** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# ttl-security`	(Optional) Enables TTL security for eBGP neighbor group nbr-grp-A.
Step 16	**update-source** *interface-type interface-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# update-source Loopback0`	(Optional) Configures the router to use the IP address from the Loopback0 interface when trying to open a session with neighbor group nbr-grp-A.
Step 17	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit`	Exits the current configuration mode.
Step 18	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

	Command or Action	Purpose
Step 19	**use neighbor-group** *group-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group nbr-grp-A`	(Optional) Specifies that BGP neighbor 172.168.40.24 inherit configuration from neighbor group nbr-grp-A.

Step 20	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a BGP Neighbor

Perform this task to configure BGP neighbors.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 5	**`address-family`** `{`**`ipv4 unicast`** `|` **`ipv4 multicast`** `|` **`ipv6 unicast`** `|` **`ipv6 multicast`**`}`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Route Reflector for BGP

Perform this task to configure a route reflector for BGP.

All the neighbors configured with the **route-reflector-client** command are members of the client group, and the remaining iBGP peers are members of the nonclient group for the local route reflector.

Together, a route reflector and its clients form a *cluster*. A cluster of clients usually has a single route reflector. In such instances, the cluster is identified by the software as the router ID of the route reflector. To increase redundancy and avoid a single point of failure in the network, a cluster can have more than one route reflector. If it does, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. The **bgp cluster-id** command is used to configure the cluster ID when the cluster has more than one route reflector.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp cluster-id** *cluster-id*
4. **neighbor** *ip-address*
5. **remote-as** *autonomous-system-number*
6. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
7. **route-reflector-client**
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`bgp cluster-id`** *`cluster-id`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp cluster-id 192.168.70.1`
		Configures the local router as one of the route reflectors serving the cluster. It is configured with the cluster ID of 192.168.70.1 to identify the cluster.

	Command or Action	Purpose
Step 4	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.
Step 5	**remote-as** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 6	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

Step 7	**route-reflector-client** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures the router as a BGP route reflector and configures the neighbor 172.168.40.24 as its client.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP Route Filtering by Route Policy

Perform this task to configure BGP routing filtering by route policy.

Prerequisites

See the *Implementing Routing Policy on Cisco IOS XR Software* module of the *Cisco IOS XR Routing Configuration Guide* for a list of the supported attributes and operations that are valid for inbound and outbound neighbor policy filtering.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **end-policy**
4. **router bgp** *autonomous-system-number*
5. **neighbor** *ip-address*
6. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
7. **route-policy** *route-policy-name* {**in** | **out**}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`route-policy`** *`name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234`
	`RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`RP/0/RP0/CPU0:router(config-rpl)# apply check-communities`
	`RP/0/RP0/CPU0:router(config-rpl)# else`
	`RP/0/RP0/CPU0:router(config-rpl)# pass`
	`RP/0/RP0/CPU0:router(config-rpl)# endif`
		(Optional) Defines a route policy named drop-as-1234 and enters route policy configuration mode.

Step 3	**`end-policy`**
	Example:
	`RP/0/RP0/CPU0:router(config-rpl)# end-policy`
		(Optional) Ends the definition of a route policy and exits route policy configuration mode.

	Command or Action	Purpose
Step 4	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 5	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 6	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

Step 7	**route-policy** *route-policy-name* {**in** | **out**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`
		Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling Next Hop Processing on BGP Updates

Perform this task to disable next hop calculation for a neighbor and insert your own address in the next hop field of BGP updates. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the network device as the next hop.

Note Next hop processing can be disabled for address family group, neighbor group, or neighbor address family.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
6. **next-hop-self**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

	Command or Action	Purpose

Step 5	**address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

Step 6	**next-hop-self** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	Sets the next hop attribute for all IPv4 unicast routes advertised to neighbor 172.168.40.24 to the address of the local router. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the local network device as the next hop.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring BGP Community and Extended-Community Filtering

Perform this task to specify that community attributes should be sent to an eBGP neighbor.

Perform this task to specify that community/extended-community attributes should be sent to an eBGP neighbor. These attributes are not sent to an eBGP neighbor by default. By contrast, they are always sent to iBGP neighbors. This section provides examples on how to enable sending community attributes. The **send-community-ebgp** keyword can be replaced by the **send-extended-community-ebgp** keyword to enable sending extended-communities.

Note If the **send-community-ebgp** command is configured for a neighbor group or address family group, all neighbors using the group inherit the configuration. Configuring the command specifically for a neighbor overrides inherited values.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
6. **send-community-ebgp**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`router bgp` *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	`neighbor` *`ip-address`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	`remote-as` *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 5	`address-family {ipv4 unicast | ipv4 multicast | ipv6 unicast | ipv6 multicast}`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

	Command or Action	Purpose

Step 6	**`send-community-ebgp`**
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-community-ebgp`
		Specifies that the router send community attributes (which are disabled by default for eBGP neighbors) to eBGP neighbor 172.168.40.24 for IPv4 multicast routes.

Step 7	**`end`**
	or
	`commit`
	
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Software to Store Updates from a Neighbor

Perform this task to configure the software to store updates received from a neighbor.

The **soft-reconfiguration inbound** command causes a route refresh request to be sent to the neighbor if the neighbor is route refresh capable. If the neighbor is not route refresh capable, the neighbor must be reset to relearn received routes using the **clear bgp soft** command. See the "Resetting Neighbors Using BGP Dynamic Inbound Soft Reset" section on page RC-71.

Note Storing updates from a neighbor works only if either the neighbor is route refresh capable or if the **soft-reconfiguration inbound** command is configured. Even if the neighbor is route refresh capable and the **soft-reconfiguration inbound** command is configured, the original routes are not stored unless the **always** option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The **soft-reconfiguration inbound** command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*

4. **address-family** {**ipv4 unicast** | **ipv4 multicast** | **ipv6 unicast** | **ipv6 multicast**}
5. **soft-reconfiguration inbound always**
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**`address-family`** {**`ipv4 unicast`** | **`ipv4 multicast`** | **`ipv6 unicast`** | **`ipv6 multicast`**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Enters neighbor address family configuration mode for the IPv4 address family.

	Command or Action	Purpose

Step 5	**`soft-reconfiguration inbound always`**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# soft-reconfiguration inbound always`
		Configures the software to store updates received from neighbor 172.168.40.24. Soft reconfiguration inbound causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a "soft clear" to be performed after the inbound policy is changed.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling a BGP Neighbor

Perform this task to administratively shut down a neighbor without removing the configuration.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **shutdown**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.
Step 4	**shutdown** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# shutdown`	Disables all active sessions for neighbor 172.168.40.24.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Resetting Neighbors Using BGP Dynamic Inbound Soft Reset

Perform this task to trigger an inbound soft reset of the specified address families for the specified group or neighbors.

Resetting neighbors is useful if you change the inbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates. If an inbound soft reset is triggered, BGP sends a REFRESH request to the neighbor if the neighbor has advertised the ROUTE_REFRESH capability. To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** {**ipv4** | **ipv6** | **all**} {**unicast** | **multicast** | **all**} {* | *ip-address* | *as-number* | **external**} **soft in**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors`	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	**clear bgp** {**ipv4** | **ipv6** | **all**} {**unicast** | **multicast** | **all**} {***** | *ip-address* | *as-number* | **external**} **soft in** **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.1 soft in`	Soft resets a BGP neighbor. • The * keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

Resetting Neighbors Using BGP Outbound Soft Reset

Perform this task to trigger an outbound soft reset of the specified address families for the specified group or neighbors.

Resetting neighbors is useful if you change the outbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates.

If an outbound soft reset is triggered, BGP resends all routes for the address family to the given neighbors.

To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** {**ipv4** | **ipv6** | **all**} {**unicast** | **multicast** | **all**} {* | *ip-address* | *as-number* | **external**} **soft out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors`	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	**clear bgp** {**ipv4** | **ipv6** | **all**} {**unicast** | **multicast** | **all**} {***** | *ip-address* | *as-number* | **external**} **soft out** **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.2 soft out`	Soft resets a BGP neighbor. • The * keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

Resetting Neighbors Using BGP Hard Reset

Perform this task to reset neighbors using a hard reset. A hard reset removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. If the **graceful** keyword is specified, the routes from the neighbor are not removed from the BGP table immediately, but are marked as stale. After the session is re-established, any stale route that has not been received again from the neighbor is removed.

SUMMARY STEPS

1. **clear bgp** {* | *ip-address* | *as-number* | **external**} [**graceful**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** {***** | *ip-address* | *as-number* | **external**} [**graceful**] **Example:** `RP/0/RP0/CPU0:router# clear bgp 10.0.0.3`	Clears a BGP neighbor. The **graceful** keyword specifies a graceful restart.

Clearing Caches, Tables and Databases

Perform this task to remove all contents of a particular cache, table, or database. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

SUMMARY STEPS

1. **clear bgp** *ip-address*
2. **clear bgp external**
3. **clear bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** *ip-address* **Example:** `RP/0/RP0/CPU0:router# clear bgp 172.20.1.1`	Clears neighbor 172.20.1.1.
Step 2	**clear bgp external** **Example:** `RP/0/RP0/CPU0:router# clear bgp external`	Clears all external peers.
Step 3	**clear bgp *** **Example:** `RP/0/RP0/CPU0:router# clear bgp *`	Clears all BGP neighbors.

Displaying System and Network Statistics

Perform this task to display specific statistics, such as the contents of BGP routing tables, caches, and databases. Information provided can be used to determine resource usage and solve network problems. You can also display information about node reachability and discover the routing path that the packets of your device are taking through the network.

SUMMARY STEPS

1. **show bgp cidr-only**
2. **show bgp count-only**
3. **show bgp community** *community-list* [**exact-match**]
4. **show bgp regexp** *regular-expression*
5. **show bgp**
6. **show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**]
7. **show bgp paths**

8. **show bgp neighbor-group** *group-name* **configuration**
9. **show bgp summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp cidr-only** **Example:** `RP/0/RP0/CPU0:router# show bgp cidr-only`	Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.
Step 2	**show bgp count-only** **Example:** `RP/0/RP0/CPU0:router# show bgp count-only`	Displays the number of paths.
Step 3	**show bgp community** *community-list* [**exact-match**] **Example:** `RP/0/RP0/CPU0:router# show bgp community 1081:5 exact-match`	Displays routes that match the BGP community 1081:5.
Step 4	**show bgp regexp** *regular-expression* **Example:** `RP/0/RP0/CPU0:router# show bgp regexp "^3 "`	Displays routes that match the autonomous system path regular expression "^3 ".
Step 5	**show bgp** **Example:** `RP/0/RP0/CPU0:router# show bgp`	Displays entries in the BGP routing table.
Step 6	**show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**] **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors 10.0.101.1`	Displays information about the BGP connection to neighbor 10.0.101.1. • The **advertised-routes** keyword displays all routes the router advertised to the neighbor. • The **dampened-routes** keyword displays the dampened routes that are learned from the neighbor. • The **flap-statistics** keyword displays flap statistics of the routes learned from the neighbor. • The **performance-statistics** keyword displays performance statistics relating to work done by the BGP process for this neighbor. • The **received** *prefix-filter* keyword and argument display the received prefix list filter. • The **routes** keyword displays routes learned from the neighbor.

	Command or Action	Purpose

Step 7	**`show bgp paths`**
	Example:
	`RP/0/RP0/CPU0:router# show bgp paths`
		Displays all BGP paths in the database.

Step 8	**`show bgp neighbor-group`** *`group-name`* **`configuration`**
	Example:
	`RP/0/RP0/CPU0:router# show bgp neighbor-group group_1 configuration`
		Displays the effective configuration for neighbor group group_1, including any configuration inherited by this neighbor group.

Step 9	**`show bgp summary`**
	Example:
	`RP/0/RP0/CPU0:router# show bgp summary`
		Displays the status of all BGP connections.

Monitoring BGP Update Groups

This task displays information related to the processing of BGP update groups.

SUMMARY STEPS

1. **show bgp** [{**ipv4** | **ipv6** | **all**} {**unicast** | **multicast** | **all**}] **update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`show bgp`** `[{`**`ipv4`** `|` **`ipv6`** `|` **`all`**`}` `{`**`unicast`** `|` **`multicast`** `|` **`all`**`}]` **`update-group`** `[`**`neighbor`** *`ip-address`* `|` *`process-id.index`* `[`**`summary`** `|` **`performance-statistics`**`]]`
	Example:
	`RP/0/RP0/CPU0:router# show bgp update-group 0.0`
		Displays information about BGP update groups.
		• The *ip-address* argument displays the update groups to which that neighbor belongs.
		• The *process-id.index* argument selects a particular update group to display and is specified as follows: process id (dot) index. Process ID range is from 0 to 254. Index range is from 0 to 4294967295.
		• The **summary** keyword displays summary information for neighbors in a particular update group.
		• If no argument is specified, this command displays information for all update groups (for the specified address family).
		• The **performance-statistics** keyword displays performance statistics for an update group.

Configuration Examples for Implementing BGP on Cisco IOS XR Software

This section provides the following configuration examples:

Enabling BGP: Example

The following shows how to enable BGP.

```
prefix-set static
   2020::/64,
   2012::/64,
   10.10.0.0/16,
   10.2.0.0/24
end-set

route-policy pass-all
  pass
end-policy
route-policy set_next_hop_agg_v4
  set next-hop 10.0.0.1
end-policy

route-policy set_next_hop_static_v4
  if (destination in static) then
    set next-hop 10.1.0.1
  else
    drop
  endif
end-policy
route-policy set_next_hop_agg_v6
  set next-hop 2003::121
end-policy
route-policy set_next_hop_static_v6
  if (destination in static) then
     set next-hop 2011::121
  else
     drop
  endif
end-policy
router bgp 65000
  bgp fast-external-fallover disable
  bgp confederation peers
    65001
    65002
  bgp confederation identifier 1
  bgp router-id 1.1.1.1
  address-family ipv4 unicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
    redistribute static route-policy set_next_hop_static_v4
  address-family ipv4 multicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
    redistribute static route-policy set_next_hop_static_v4
  address-family ipv6 unicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6
  address-family ipv6 multicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6
  neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast
    address-family ipv4 multicast
  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast
    address-family ipv4 multicast
  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    address-family ipv4 multicast
      route-policy pass-all in
      route-policy pass-all out
  neighbor 10.0.101.64
    remote-as 5
    update-source Loopback0
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    address-family ipv4 multicast
      route-policy pass-all in
      route-policy pass-all out
```

Displaying BGP Update Groups: Example

The following is sample output from the **show bgp update-group** command executed in EXEC mode:

```
RP/0/RP0/CPU0:router# show bgp update-group

Update group for IPv4 Unicast, index 0.1:
  Attributes:
    Outbound Route map:rm
    Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.92

Update group for IPv4 Unicast, index 0.2:
  Attributes:
    Minimum advertisement interval:30
  Messages formatted:2, replicated:2
  Neighbors in this update group:
    10.0.101.91
```

BGP Neighbor Configuration: Example

The following example shows how BGP neighbors on an autonomous system are configured to share information. In the example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured shares information about networks 131.108.0.0 and 192.31.7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second **neighbor** and **remote-as** commands specify an internal neighbor (with the same autonomous system number) at address 131.108.234.2; and the third **neighbor** and **remote-as** commands specify a neighbor on a different autonomous system.

```
router bgp 109
 network 131.108.0.0
 network 192.31.7.0
 neighbor 131.108.200.1
  remote-as 167
 neighbor 131.108.234.2
  remote-as 109
 neighbor 150.136.64.19
  remote-as 99
```

BGP Confederation: Example

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified using the **bgp confederation identifier** command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence, peers 171.69.232.55 and 171.69.232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 160.69.69.1 is a normal eBGP speaker and the updates received by it from this peer are just like a normal eBGP update from a peer in autonomous system 666.

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers 6002 6003
 neighbor 171.69.232.55
 remote-as 6002
 neighbor 171.69.232.56
 remote-as 6003
 neighbor 160.69.69.1
 remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. Peer 170.70.70.1 is a normal iBGP peer and peer 199.99.99.2 is a normal eBGP peer from autonomous system 700.

```
router bgp 6002
 bgp confederation identifier 666
 bgp confederation peers 6001 6003
 neighbor 170.70.70.1
  remote-as 6002
 neighbor 171.69.232.57
  remote-as 6001
 neighbor 171.69.232.56
  remote-as 6003
 neighbor 199.99.99.2
  remote-as 700
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. Peer 200.200.200.200 is a normal eBGP peer from autonomous system 701.

```
router bgp 6003
 bgp confederation identifier 666
 bgp confederation peers 6001 6002
 neighbor 171.69.232.57
  remote-as 6001
 neighbor 171.69.232.55
  remote-as 6002
 neighbor 200.200.200.200
  remote-as 701
```

The following is a part of the configuration from the BGP speaker 200.200.200.205 from autonomous system 701 in the same example. Neighbor 171.69.232.56 is configured as a normal eBGP speaker from autonomous system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
 neighbor 171.69.232.56
  remote-as 666
 neighbor 200.200.200.205
  remote-as 701
```

BGP Route Reflector: Example

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client for both unicast and multicast prefixes:

```
router bgp 140
 neighbor 10.1.1.1
  remote-as 140
  address-family ipv4 unicast
   route-reflector-client

router bgp 140
 neighbor 10.1.1.1
  remote-as 140
  address-family ipv4 multicast
   route-reflector-client
```

Where to Go Next

For detailed information about BGP commands, see the *Cisco IOS XR Routing Command Reference* document.

Additional References

The following sections provide references related to implementing BGP for Cisco IOS XR software.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*, Release 3.2

Standards

Standards	Title
draft-ietf-idr-bgp4-26.txt	*A Border Gateway Protocol 4*, by Y. Rekhter, T.Li, S. Hares
draft-ietf-idr-bgp4-mib-15.txt	*Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4)*, by J. Hass and S. Hares
draft-ietf-idr-cease-subcode-05.txt	*Subcodes for BGP Cease Notification Message*, by Enke Chen, V. Gillet

MIBs

MIBs	MIBs Link
• BGP4-MIB • CISCO-BGP4-MIB	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1997	*BGP Communities Attribute*
RFC 2385	*Protection of BGP Sessions via the TCP MD5 Signature Option*
RFC 2439	*BGP Route Flap Damping*
RFC 2545	*Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*
RFC 2796	*BGP Route Reflection - An Alternative to Full Mesh IBGP*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*

RFCs	Title
RFC 3065	*Autonomous System Confederations for BGP*
RFC 3392	*Capabilities Advertisement with BGP-4*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing IS-IS on Cisco IOS XR Software

Integrated Intermediate System-to-Intermediate System (IS-IS), Internet Protocol Version 4 (IPv4), is a standards-based Interior Gateway Protocol (IGP).

Cisco IOS XR implements the IP routing capabilities described in International Organization for Standardization (ISO)/International Engineering Consortium (IEC) 10589 and RFC 1995, and adds the standard extensions for single topology and multitopology IS-IS for IP Version 6 (IPv6).

This module describes the new and revised tasks you need to implement IS-IS (IPv4 and IPv6) on your Cisco IOS XR network.

Note For more information about IS-IS on the Cisco IOS XR software and complete descriptions of the IS-IS commands listed in this module, you can refer to the "Related Documents" section of this module. To locate documentation for other commands that might appear while of executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing IS-IS on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router. The ability to configure a broadcast medium connecting two networking devices as a point-to-point link was added.
Release 3.2.2	Support was added for the multicast-intact feature.

Contents

Prerequisites for Implementing IS-IS on Cisco IOS XR Software

The following are prerequisites for implementing IS-IS on Cisco IOS XR software:

- You must be in a user group associated with a task group that includes the proper task IDs for IS-IS commands. Task IDs for commands are listed in the Cisco IOS XR Task ID Reference Guide. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.

Restrictions for Implementing IS-IS on Cisco IOS XR Software

When multiple instances of IS-IS are being run, an interface can be associated with only one instance (process). Instances may not share an interface. Additionally, if Multiprotocol Label Switching traffic engineering (MPLS TE) is being employed, then MPLS must be enabled for only one instance. The MPLS process is not multi-instance aware.

Information About Implementing IS-IS on Cisco IOS XR Software

To implement IS-IS you need to understand the following concepts:

- IS-IS Functional Overview, page RC-85
- Key Features Supported in the Cisco IOS XR IS-IS Implementation, page RC-85
- IS-IS Configuration Grouping, page RC-85
- Multitopology Configuration, page RC-86
- IPv6 Routing and Configuring IPv6 Addressing, page RC-86
- Limit LSP Flooding, page RC-86
- Maximum LSP Lifetime and Refresh Interval, page RC-87
- Overload Bit Configuration During Multitopology Operation, page RC-87
- Single-Topology IPv6 Support, page RC-87
- Multitopology IPv6 Support, page RC-88
- Nonstop Forwarding, page RC-88
- Multi-Instance IS-IS, page RC-89
- Multiprotocol Label Switching Traffic Engineering, page RC-89
- Overload Bit on Router, page RC-89
- Default Routes, page RC-90
- Attached Bit on an IS-IS Instance, page RC-90

IS-IS Functional Overview

Small IS-IS networks are typically built as a single area that includes all routers in the network. As the network grows larger, it may be reorganized into a backbone area made up of the connected set of all Level 2 routers from all areas, which is in turn connected to local areas. Within a local area, routers know how to reach all system IDs. Between areas, routers know how to reach the backbone, and the backbone routers know how to reach other areas.

The IS-IS routing protocol supports the configuration of backbone Level 2 and Level 1 areas and the necessary support for moving routing information between the areas. Routers establish Level 1 adjacencies to perform routing within a local area (intra-area routing). Routers establish Level 2 adjacencies to perform routing between Level 1 areas (interarea routing).

For Cisco IOS XR software, each IS-IS instance can support either a single Level 1 or Level 2 area, or one of each. By default, all IS-IS instances automatically support Level 1 and Level 2 routing. You can change the level of routing to be performed by a particular routing instance using the **is-type** command.

Key Features Supported in the Cisco IOS XR IS-IS Implementation

The Cisco IOS XR implementation of IS-IS conforms to the IS-IS Version 2 specifications detailed in RFC 1195 and the IPv6 IS-IS functionality based on the Internet Engineering Task Force (IETF) IS-IS Working Group draft-ietf-isis-ipv6.txt document.

The following list outlines key features supported in the Cisco IOS XR implementation:

- Improved configuration syntax and enhanced **show** commands
- Single topology IPv6
- Multitopology
- Nonstop forwarding (NSF), both Cisco proprietary and IETF
- Three-way handshake
- Mesh groups
- Multiple IS-IS instances
- Configuration of a broadcast medium connecting two networking devices as a point-to-point link

IS-IS Configuration Grouping

Cisco IOS XR groups all of the IS-IS configuration in router configuration mode, including the portion of the interface configurations associated with IS-IS. The grouping makes the configuration process clearer, and eliminates some of the clutter in the global interface stanza. To display the IS-IS configuration in its entirety, use the **show isis interface** command.

The command output displays the running configuration for all configured IS-IS instances, including the interface assignments and interface attributes.

IS-IS Interfaces

IS-IS interfaces can be configured as one of the following types:

- active—advertises connected prefixes and forms adjacencies. This is the default for interfaces.
- passive—advertises connected prefixes but does not form adjacencies. The **passive** command is used to configure interfaces as passive. Passive interfaces should be used sparingly for important prefixes such as loopback addresses that need to be injected into the IS-IS domain. If many connected prefixes need to be advertised then the redistribution of connected routes with the appropriate policy should be used instead.
- suppressed—does not advertise connected prefixes but forms adjacencies. The **suppress** command is used to configure interfaces as suppressed.
- shutdown—does not advertise connected prefixes and does not form adjacencies. The **shutdown** command is used to disable interfaces without removing the IS-IS configuration.

Multitopology Configuration

Cisco IOS XR software supports multitopology for IPv6 IS-IS unless single topology is explicitly configured in IPv6 address-family configuration mode.

Note IS-IS supports IP routing and not Open Systems Interconnection (OSI) Connectionless Network Service (CLNS) routing.

IPv6 Routing and Configuring IPv6 Addressing

By default, IPv6 routing is disabled in the Cisco IOS XR software. To enable IPv6 routing, you must assign IPv6 addresses to individual interfaces in the router using the **ipv6 enable** or **ipv6 address** command. See the *Network Stack IPv4 and IPv6 Commands on Cisco IOS XR Software* module of the *Cisco IOS XR IP Addresses and Services Command Reference*.

Limit LSP Flooding

Limiting link-state packets (LSP) may be desirable in certain “meshy” network topologies. An example of such a network might be a highly redundant one such as a fully meshed set of point-to-point links over a nonbroadcast multiaccess (NBMA) transport. In such networks, full LSP flooding can limit network scalability. One way to restrict the size of the flooding domain is to introduce hierarchy by using multiple Level 1 areas and a Level 2 area. However, two other techniques can be used instead of or with hierarchy: Block flooding on specific interfaces and configure mesh groups.

Both techniques operate by restricting the flooding of LSPs in some fashion. A direct consequence is that although scalability of the network is improved, the reliability of the network (in the face of failures) is reduced because a series of failures may prevent LSPs from being flooded throughout the network, even though links exist that would allow flooding if blocking or mesh groups had not restricted their use. In such a case, the link-state databases of different routers in the network may no longer be synchronized. Consequences such as persistent forwarding loops can ensue. For this reason, we recommend that blocking or mesh groups be used only if specifically required, and then only after careful network design.

Flood Blocking on Specific Interfaces

With this technique, certain interfaces are blocked from being used for flooding LSPs, but the remaining interfaces operate normally for flooding. This technique is simple to understand and configure, but may be more difficult to maintain and more error prone than mesh groups in the long run. The flooding topology that IS-IS uses is fine-tuned rather than restricted. Restricting the topology too much (blocking too many interfaces) makes the network unreliable in the face of failures. Restricting the topology too little (blocking too few interfaces) may fail to achieve the desired scalability.

To improve the robustness of the network in the event that all nonblocked interfaces drop, use the **csnp-interval** command in interface configuration mode to force periodic complete sequence number PDUs (CSNPs) packets to be used on blocked point-to-point links. The use of periodic CSNPs enables the network to become synchronized.

Mesh Group Configuration

Configuring mesh groups (a set of interfaces on a router) can help to limit flooding. All routers reachable over the interfaces in a particular mesh group are assumed to be densely connected with each router having at least one link to every other router. Many links can fail without isolating one or more routers from the network.

In normal flooding, a new LSP is received on an interface and is flooded out over all other interfaces on the router. With mesh groups, when a new LSP is received over an interface that is part of a mesh group, the new LSP is not   flooded over the other interfaces that are part of that mesh group.

Maximum LSP Lifetime and Refresh Interval

By default, the router sends a periodic LSP refresh every 15 minutes. LSPs remain in a database for 20 minutes by default. If they are not refreshed by that time, they are deleted. You can change the LSP refresh interval or maximum LSP lifetime. The LSP interval should be less than the LSP lifetime or else LSPs time out before they are refreshed. In the absence of a configured refresh interval, the software adjusts the LSP refresh interval, if necessary, to prevent the LSPs from timing out.

Overload Bit Configuration During Multitopology Operation

Because the overload bit applies to forwarding for a single topology, it may be configured and cleared independently for IPv4 and IPv6 during multitopology operation. For this reason, the overload is set from the router address family configuration mode. If the IPv4 overload bit is set, all routers in the area do not use the router for IPv4 transit traffic. However, they can still use the router for IPv6 transit traffic.

Single-Topology IPv6 Support

Single-topology IPv6 support on Cisco IOS XR software allows IS-IS for IPv6 to be configured on interfaces along with an IPv4 network protocol. All interfaces must be configured with the identical set of network protocols, and all routers in the IS-IS area (for Level 1 routing) or the domain (for Level 2 routing) must support the identical set of network layer protocols on all interfaces.

When single-topology support for IPv6 is used, only narrow link metrics, also known as old-style type, length, and value (TLV) arguments, may be employed. During single-topology operation, one shortest path first (SPF) computation for each level is used to compute both IPv4 and IPv6 routes. Using a single SPF is possible because both IPv4 IS-IS and IPv6 IS-IS routing protocols share a common link topology.

Because multitopology is the default behavior in the software, you must explicitly configure IPv6 to use the same topology as IPv4 enable single-topology IPv6. Configure the **single-topology** command in IPv6 router address family configuration submode of the IS-IS router stanza.

Multitopology IPv6 Support

Multitopology IPv6 support on Cisco IOS XR software for IS-IS assumes that multitopology support is required as soon as it detects interfaces configured for both IPv6 and IPv4 within the IS-IS stanza.

Nonstop Forwarding

On Cisco IOS XR software, NSF minimizes the amount of time a network is unavailable to its users following a route processor (RP) failover. The main objective of NSF is to continue forwarding IP packets and perform a graceful restart following an RP failover.

When a router restarts, all routing peers of that device usually detect that the device went down and then came back up. This transition results in what is called a *routing flap*, which could spread across multiple routing domains. Routing flaps caused by routing restarts create routing instabilities, which are detrimental to the overall network performance. NSF helps to suppress routing flaps in NSF-aware devices, thus reducing network instability.

NSF allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following an RP failover. When the NSF feature is configured, peer networking devices do not experience routing flaps. Data traffic is forwarded through intelligent line cards or dual forwarding processors (FPs) while the standby RP assumes control from the failed active RP during a failover. The ability of line cards and FPs to remain up through a failover and to be kept current with the Forwarding Information Base (FIB) on the active RP is key to NSF operation.

When the Cisco IOS XR router running IS-IS routing performs an RP failover, the router must perform two tasks to resynchronize its link-state database with its IS-IS neighbors. First, it must relearn the available IS-IS neighbors on the network without causing a reset of the neighbor relationship. Second, it must reacquire the contents of the link-state database for the network.

The IS-IS NSF feature offers two options when configuring NSF:

- IETF NSF
- Cisco NSF

If neighbor routers on a network segment are NSF aware, meaning that neighbor routers are running a software version that supports the IETF Internet draft for router restartability, they assist an IETF NSF router that is restarting. With IETF NSF, neighbor routers provide adjacency and link-state information to help rebuild the routing information following a failover.

In Cisco IOS XR software, Cisco NSF checkpoints (stores persistently) all the state necessary to recover from a restart without requiring any special cooperation from neighboring routers. The state is recovered from the neighboring routers, but only using the standard features of the IS-IS routing protocol. This capability makes Cisco NSF suitable for use in networks in which other routers have not used the IETF standard implementation of NSF.

Note If you configure IETF NSF on the Cisco IOS XR router and a neighbor router does not support IETF NSF, the affected adjacencies flap, but nonstop forwarding is maintained to all neighbors that do support IETF NSF. A restart reverts to a cold start if no neighbors support IETF NSF.

Multi-Instance IS-IS

You may configure as many IS-IS instances as system resources (memory and interfaces) allow. Each interface may be associated with only a single IS-IS instance, and MPLS may be enabled for only a single IS-IS instance. Cisco IOS XR software prevents the double-booking of an interface by two instances at configuration time—two instances of MPLS configuration causes an error.

Because the Routing Information Base (RIB) treats each of the IS-IS instances as equal routing clients, you must be careful when redistributing routes between IS-IS instances. The RIB does not know to prefer Level 1 routes over Level 2 routes. For this reason, if you are running Level 1 and Level 2 instances, you must enforce the preference by configuring different administrative distances for the two instances.

Multiprotocol Label Switching Traffic Engineering

The MPLS TE feature enables an MPLS backbone to replicate and expand the traffic engineering capabilities of Layer 2 ATM and Frame Relay networks. MPLS is an integration of Layer 2 and Layer 3 technologies.

For IS-IS, MPLS TE automatically establishes and maintains MPLS TE label-switched paths across the backbone by using Resource Reservation Protocol (RSVP). The route that a label-switched path uses is determined by the label-switched paths resource requirements and network resources, such as bandwidth. Available resources are flooded by using special IS-IS TLV extensions in the IS-IS. The label-switched paths are explicit routes and are referred to as traffic engineering (TE) tunnels.

Overload Bit on Router

The overload bit is a special bit of state information that is included in an LSP of the router. If the bit is set on the router, it notifies routers in the area that the router is not available for transit traffic. This capability is useful in four situations:

1. During a serious but nonfatal error, such as limited memory.
2. During the startup and restart of the process. The overload bit can be set until the routing protocol has converged. However, it is not employed during a normal NSF restart or failover because doing so causes a routing flap.
3. During a trial deployment of a new router. The overload bit can be set until deployment is verified, then cleared.
4. During the shutdown of a router. The overload bit can be set to remove the router from the topology before the router is removed from service.

Default Routes

You can force a default route into an IS-IS routing domain. Whenever you specifically configure redistribution of routes into an IS-IS routing domain, the Cisco IOS XR software does not, by default, redistribute the default route into the IS-IS routing domain. The **default-information originate** command generates a *default route* into IS-IS, which can be controlled by a route map. You can use the route map to identify the level into which the default route is to be announced, and you can specify other filtering options configurable under a route map. You can use a route map to conditionally advertise the default route, depending on the existence of another route in the routing table of the router.

Attached Bit on an IS-IS Instance

The attached bit is set in a router that is configured with the **is-type** command and **level-1-2** keyword. The attached bit indicates that the router is connected to other areas (typically through the backbone). This functionality means that the router can be used by Level 1 routers in the area as the default route to the backbone. The attached bit is usually set automatically as the router discovers other areas while computing its Level 2 SPF route. The bit is automatically cleared when the router becomes detached from the backbone. To simulate this behavior when using multiple processes to represent the **level-1-2** keyword functionality, you would manually configure the attached bit on the Level 1 process.

Caution If the connectivity for the Level 2 instance is lost, the attached bit in the Level 1 instance LSP would continue sending traffic to the Level 2 instance and cause the traffic to be dropped.

Multicast-Intact Feature

The multicast-intact feature provides the ability to run multicast routing (PIM) when IGP shortcuts are configured and active on the router. Both OSPFv2 and IS-IS support the multicast-intact feature.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGPs route the IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel (from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next-hops for use by PIM. These next-hops are called *mcast-intact* next-hops. The mcast-intact next-hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next-hop to a PIM source.
- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost mcast-intact next-hops to the RIB. This attribute applies even when the native next-hops have no IGP shortcuts.
- In IS-IS, the max-paths limit is applied by counting both the native and mcast-intact next-hops together. (In OSPFv2, the behavior is slightly different.)

How to Implement IS-IS on Cisco IOS XR Software

This section contains the following procedures:

- Enabling IS-IS and Configuring Level 1 or Level 2 Routing, page RC-91 (required)
- Configuring Single Topology for IS-IS, page RC-93 (required)
- Configuring Multitopology for IS-IS, page RC-98 (optional)
- Controlling LSP Flooding for IS-IS, page RC-102 (optional)
- Configuring Nonstop Forwarding for IS-IS, page RC-106 (optional)
- Configuring Authentication for IS-IS, page RC-108 (optional)
- Configuring MPLS Traffic Engineering for IS-IS, page RC-110 (optional)
- Tuning Adjacencies for IS-IS on Point-to-Point Interfaces, page RC-113 (optional)
- Setting SPF Interval for a Single-Topology IPv4 and IPv6 Configuration, page RC-116 (optional)
- Enabling Multicast-Intact for IS-IS, page RC-118 (optional)

Note To save configuration changes, you must commit changes when the system prompts you.

Enabling IS-IS and Configuring Level 1 or Level 2 Routing

This task explains how to enable IS-IS and configure the routing level for an area.

Note Configuring the routing level in Step 4 is optional, but is highly recommended to establish the proper level of adjacencies.

Prerequisites

Although you can configure IS-IS before you configure an IP address, no IS-IS routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **net** *network-entity-title*
4. **is-type** {**level-1** | **level-1-2** | **level-2-only**}
5. **end**
   or
   commit
6. **show isis** [**instance** *instance-id*] **protocol**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. • By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.
Step 3	**net** *network-entity-title* **Example:** `RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00`	Configures network entity titles (NETs) for the routing instance. • Specify a NET for each routing instance if you are configuring multi-instance IS-IS. You can specify a name for a NET and for an address. • This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00. • To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the systemID portion of the NET must match exactly for all of the configured items.

Step 4	**is-type** {**level-1** | **level-1-2** | **level-2-only**}
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# is-type level-2-only`
		(Optional) Configures the system type (area or backbone router).
		• By default, every IS-IS instance acts as a **level-1-2** router.
		• The **level-1** keyword configures the software to perform Level 1 (intra-area) routing only. Only Level 1 adjacencies are established. The software learns about destinations inside its area only. Any packets containing destinations outside the area are sent to the nearest **level-1-2** router in the area.
		• The **level-2-only** keyword configures the software to perform Level 2 (backbone) routing only, and the router establishes only Level 2 adjacencies, either with other Level 2-only routers or with **level-1-2** routers.
		• The **level-1-2** keyword configures the software to perform both Level 1 and Level 2 routing. Both Level 1 and Level 2 adjacencies are established. The router acts as a border router between the Level 2 backbone and its Level 1 area.

	Command or Action	Purpose

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 6	**`show isis`** `[`**`instance`** *`instance-id`*`]` **`protocol`**
	Example:
	`RP/0/RP0/CPU0:router# show isis protocol`
		(Optional) Displays summary information about the IS-IS instance.

Configuring Single Topology for IS-IS

After an IS-IS instance is enabled, it must be configured to compute routes for a specific network topology.

This task explains how to configure the operation of the IS-IS protocol on an interface for an IPv4 or IPv6 topology.

Restrictions

To enable the router to run in single-topology mode, configure each of the IS-IS interfaces with all of the address families enabled and “single-topology” in the address-family IPv6 unicast in the IS-IS router stanza. You can use either the IPv6 address family or both IPv4 and IPv6 address families, but your configuration must represent the set of all active address families on the router. In addition, you should explicitly enable single-topology operation by configuring it in the IPv6 router address family submode. Exceptions to these instructions exist:

1. If the address-family stanza in the IS-IS process contains the **adjacency-check disable** command, then an interface is not required to have the address family enabled.
2. If the interface is configured to Level 2 only. (This exception permits the running of IPv4 and IPv6 areas.)
3. The **single-topology** command is not valid in the ipv4 address-family submode.

The default metric style for single topology is narrow metrics. However, you can use either wide metrics or narrow metrics. How to configure them depends on how single topology is configured. If both IPv4 and IPv6 are enabled and single topology is configured, the metric style is configured in the **address-family ipv4** stanza. You may configure the metric style in the **address-family ipv6** stanza, but it is ignored in this case. If only IPv6 is enabled and single topology is configured, then the metric style is configured in the **address-family ipv6** stanza.

SUMMARY STEPS

1. **configure**
2. **interface** *type number*
3. **ipv4 address** *address mask*
   or
   ipv6 address *ipv6-prefix*/*prefix-length* [**eui-64**]
   or
   ipv6 address *ipv6-address* {/*prefix-length* | *link-local*}
   or
   ipv6 enable
4. **exit**
5. **router isis** *instance-id*
6. **net** *network-entity-title*
7. **address-family ipv6** [**unicast**]
8. **single-topology**
9. **exit**
10. **interface** *type instance*
11. **circuit-type** {**level-1** | **level-1-2** | **level-2-only**}
12. **address-family** {**ipv4** | **ipv6**} [**unicast**]
13. **end**
    or
    commit
14. **show isis** [**instance** *instance-id*] **interface** [*type instance*] [**detail**] [**level** {**1** | **2**}]
15. **show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** {**1** | **2**}] [**summary**]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type number`*
	Example:
	`RP/0/RP0/CPU0:router(config)# interface POS 0/1/0/3`
		Enters interface configuration mode.

Step 3	**`ipv4 address`** *`address mask`*
	or
	`ipv6 address` *`ipv6-prefix`***`/`***`prefix-length`* [**`eui-64`**]
	or
	`ipv6 address` *`ipv6-address`* {**`/`***`prefix-length`* | *`link-local`*}
	or
	`ipv6 enable`
	Example:
	`RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 enable`
		Defines the IPv4 address for the interface. An IP address is required on all interfaces in an area enabled for IS-IS if any one interface is configured for IS-IS routing.
		or
		Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface with the **eui-64** keyword.
		or
		Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface with the **link-local** keyword.
		or
		Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing.
		• The link-local address can be used only to communicate with nodes on the same link.
		• Specifying the **ipv6 address** *ipv6-prefix***/***prefix-length* interface configuration command without the **eui-64** keyword configures site-local and global IPv6 addresses.
		• Specifying the **ipv6 address** *ipv6-prefix***/***prefix-length* command with the **eui-64** keyword configures site-local and global IPv6 addresses with an interface ID in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.
		• Specifying the **ipv6 address** command with the **link-local** keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.

	Command or Action	Purpose

Step 4	**`exit`**
	Example:
	`RP/0/RP0/CPU0:router(config-if)# exit`
		Exits interface configuration mode, and returns the router to global configuration mode.

Step 5	**`router isis`** *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
		• By default, all IS-IS instances are Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance using the **is-type** command.

Step 6	**`net`** *`network-entity-title`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00`
		Configures NETs for the routing instance.
		• Specify a NET for each routing instance if you are configuring multi-instance IS-IS. You can specify a name for a NET and for an address.
		• This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.00.
		• To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the system ID portion of the NET must match exactly for all of the configured items.

Step 7	**`address-family ipv6`** `[`**`unicast`**`]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast`
		Specifies the IPv6 address family and enters router address family configuration mode.
		• This example specifies the unicast IPv6 address family.

Step 8	**`single-topology`**
	Example:
	`RP0/0/RP0/CPU0:router(config-isis-af)# single-topology`
		(Optional) Configures the link topology for IPv4 when IPv6 is configured.
		• The **single-topology** command is valid only in IPv6 submode. The command instructs IPv6 to use the single topology rather than the default configuration of a separate topology in the multitopology mode.
		• See the "Single-Topology IPv6 Support" section on page RC-87 for more information.

Step 9	**`exit`**
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# exit`
		Exits router address family configuration mode, and returns the router to router configuration mode.

Step 10	**`interface`** *`type instance`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# interface POS 0/1/0/3`
		Enters interface configuration mode.

	Command or Action	Purpose

Step 11	**circuit-type** {**level-1** | **level-1-2** | **level-2-only**}
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# circuit-type level-1-2`
		(Optional) Configures the type of adjacency.
		• The default circuit type is the configured system type (configured through the **is-type** command).
		• Typically, the circuit type must be configured when the router is configured as only **level-1-2** and you want to constrain an interface to form only **level-1** or **level-2-only** adjacencies.

Step 12	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv6 unicast`	Specifies the IPv4 or IPv6 address family, and enters interface address family configuration mode. • This example specifies the unicast IPv6 address family on the interface.

Step 13	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 14	**show isis** [**instance** *instance-id*] **interface** [*type instance*] [**detail**] [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router# show isis interface POS0/1/0/1`	(Optional) Displays information about the IS-IS interface.
Step 15	**show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** {**1** | **2**}] [**summary**] **Example:** `RP/0/RP0/CPU0:router# show isis topology`	(Optional) Displays a list of connected routers in all areas.

Configuring Multitopology for IS-IS

This task explains how to configure multitopology IS-IS. This task is optional. Multitopology is configured in much the same way as the single topology for IPv4 and IPv6 address families. The **single-topology** command is omitted, invoking the default multitopology behavior.

SUMMARY STEPS

1. **configure**
2. **interface** *type instance*
3. **ipv4 address** *address mask*
   or
   ipv6 address *ipv6-prefix***/***prefix-length* [**eui-64**]
   or
   ipv6 address *ipv6-address* {**/***prefix-length* | *link-local*}
   or
   ipv6 enable
4. **exit**
5. **router isis** *instance-id*
6. **net** *network-entity-title*
7. **interface** *type instance*
8. **address-family ipv4** [**unicast**]
9. **exit**
10. **address-family ipv6** [**unicast**]
11. **end**
    or
    commit
12. **show isis** [**instance** *instance-id*] **interface** [*type number*] [**brief** | **detail**] [**level** {**1** | **2**}]
13. **show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** {**1** | **2**}] [**ipv4** | **ipv6**] [**summary**] [**unicast**]
14. **show isis** [**instance** *instance-id*] **adjacency** [**level** {**1** | **2**}] [*interface-type interface-instance*] [**detail**] [**systemid** *system-id*]
15. **show isis adjacency-log** [**level** {**1** | **2**}]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS 0/1/0/3`	Enters interface configuration mode.

Step 3	**ipv4 address** *address mask*
	or
	ipv6 address *ipv6-prefix***/***prefix-length* [**eui-64**]
	or
	ipv6 address *ipv6-address* {**/***prefix-length* | *link-local*}
	or
	ipv6 enable
	Example:
	`RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.0.1.3 255.255.255.0`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address 3ffe:1234:c18:1::/64 eui-64`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 address FE80::260:3EFF:FE11:6770 link-local`
	or
	`RP/0/RP0/CPU0:router(config-if)# ipv6 enable`
		Defines the IPv4 address for the interface.
		• An IP address is required on all interfaces in an area enabled for IS-IS if any one interface is configured for IS-IS routing.
		or
		Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface.
		or
		Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface.
		or
		Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing.
		• The link-local address can be used to communicate only with nodes on the same link.
		• Specifying the **ipv6 address** *ipv6-prefix***/***prefix-length* interface configuration command without the **eui-64** keyword configures site-local and global IPv6 addresses.
		• Specifying the **ipv6 address** *ipv6-prefix***/***prefix-length* command with the **eui-64** keyword configures site-local and global IPv6 addresses with an interface ID in the low-order 64 bits of the IPv6 address. Only the 64-bit network prefix for the address needs to be specified; the last 64 bits are automatically computed from the interface ID.
		• Specifying the **ipv6 address** command with the **link-local** keyword configures a link-local address on the interface that is used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface.

	Command or Action	Purpose

Step 4	**`exit`**
	Example:
	`RP/0/RP0/CPU0:router(config-if)# exit`
		Exits interface configuration mode, and returns the router to global configuration mode.

Step 5	**`router isis`** *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
		• You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.

Step 6	**`net`** *`network-entity-title`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# net 47.0004.004d.0001.0001.0c11.1110.00`
		Configures NETs for the routing instance.
		• Specify a NET for each routing instance if you are configuring multi-instance IS-IS. You can specify a name for a NET and for an address.
		• This example configures a router with area ID 47.0004.004d.0001 and system ID 0001.0c11.1110.10.
		• To specify more than one area address, specify additional NETs. Although the area address portion of the NET differs, the system ID portion of the NET must match exactly for all of the configured items.

Step 7	**`interface`** *`type instance`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# interface POS 0/1/0/4`
		Enters interface configuration mode.

Step 8	**`address-family ipv4`** [**`unicast`**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters interface address family configuration mode. • This example specifies the unicast IPv4 address family.

Step 9	**`exit`**
	Example:
	`RP/0/RP0/CPU0:router(config-if)# exit`
		Exits interface configuration mode, and returns the router to interface configuration mode.

Step 10	**`address-family ipv6`** [**`unicast`**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv6 unicast`	Specifies the IPv6 address family and enters interface address family configuration mode. • This example specifies the unicast IPv6 address family.

	Command or Action	Purpose

Step 11	**end**
	or
	commit
	Example:
	RP/0/RP0/CPU0:router(config-isis-if-af)# end
	or
	RP/0/RP0/CPU0:router(config-isis-if-af)# commit
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 12	**show isis** [**instance** *instance-id*] **interface** [*type instance*] [**brief** | **detail**] [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router# show isis interface POS 0/1/0/1 brief	(Optional) Displays information about the IS-IS interface.
Step 13	**show isis** [**instance** *instance-id*] **topology** [**systemid** *system-id*] [**level** {**1** | **2**}] [**ipv4** | **ipv6**] [**summary**] [**unicast**] **Example:** RP/0/RP0/CPU0:router# show isis topology	(Optional) Displays a list of connected routers in all areas.
Step 14	**show isis** [**instance** *instance-id*] **adjacency** [**level** {**1** | **2**}] [*interface-type interface-instance*] [***detail***] [**systemid** *system-id*] **Example::** RP/0/RP0/CPU0:router# show isis adjacency	(Optional) Displays state information about established adjacencies.
Step 15	**show isis adjacency-log** [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router# show isis adjacency-log level 1	(Optional) Displays the history of recent adjacency state transitions.

Controlling LSP Flooding for IS-IS

Flooding of LSPs can limit network scalability. You can control LSP flooding by tuning your LSP database parameters on the router globally or on the interface. This task is optional.

Many of the commands to control LSP flooding contain an option to specify the level to which they apply. Without the option, the command applies to both levels. If an option is configured for one level, the other level continues to use the default value. To configure options for both levels, use the command twice. For example:

```
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 1200 level 2
RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 1100 level 1
```

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **lsp-refresh-interval** *seconds* [**level** {**1** | **2**}]
4. **lsp-check-interval** *seconds* [**level** {**1** | **2**}]
5. **lsp-gen-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...}[**level** {**1** | **2**}]
6. **lsp-mtu** *bytes* [**level** {**1** | **2**}]
7. **max-lsp-lifetime** *seconds* [**level** {**1** | **2**}]
8. **ignore-lsp-errors disable**
9. **interface** *type instance*
10. **lsp-interval** *milliseconds* [**level** {**1** | **2**}]
11. **csnp-interval** *seconds* [**level** {**1** | **2**}]
12. **retransmit-interval** *seconds* [**level** {**1** | **2**}]
13. **retransmit-throttle-interval** *milliseconds* [**level** {**1** | **2**}]
14. **mesh-group** {*number* | **blocked**}
15. **end**
    or
    commit
16. **show isis interface [***type instance* | **level** {**1** | **2**}] [**brief**]
17. **show isis** [**instance** *instance-id*] **database** [**level** {**1** | **2**}] [**detail** | **summary** | **verbose**] [***** | *lsp-id*]
18. **show isis** [**instance** *instance-id*] **lsp-log** [**level** {**1** | **2**}]
19. **show isis database-log** [**level** {**1** | **2**}]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	**router isis** *instance-id* **Example:** RP/0/RP0/CPU0:router(config)# router isis isp	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. • You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.
Step 3	**lsp-refresh-interval** *seconds* [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router(config-isis)# lsp-refresh-interval 10800	(Optional) Sets the time between regeneration of LSPs that contain different sequence numbers • The refresh interval should always be set lower than the **max-lsp-lifetime** command.
Step 4	**lsp-check-interval** *seconds* [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router(config-isis)# lsp-check-interval 240	(Optional) Configures the time between periodic checks of the entire database to validate the checksums of the LSPs in the database. • This operation is costly in terms of CPU and so should be configured to occur infrequently.
Step 5	**lsp-gen-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...}[**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router(config-isis)# lsp-gen-interval maximum-wait 15 initial-wait 5	(Optional) Reduces the rate of LSP generation during periods of instability in the network. Helps reduce the CPU load on the router and number of LSP transmissions to its IS-IS neighbors. • During prolonged periods of network instability, repeated recalculation of LSPs can cause an increased CPU load on the local router. Further, the flooding of these recalculated LSPs to the other Intermediate Systems in the network causes increased traffic and can result in other routers having to spend more time running route calculations.
Step 6	**lsp-mtu** *bytes* [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router(config-isis)# lsp-mtu 1300	(Optional) Sets the maximum transmission unit (MTU) size of LSPs.
Step 7	**max-lsp-lifetime** *seconds* [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router(config-isis)# max-lsp-lifetime 11000	(Optional) Sets the initial lifetime given to an LSP originated by the router. • This is the amount of time that the LSP persists in the database of a neighbor unless the LSP is regenerated or refreshed.

	Command or Action	Purpose

Step 8	**`ignore-lsp-errors disable`**
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# ignore-lsp-errors disable`
		(Optional) Sets the router to purge LSPs received with checksum errors.

Step 9	**`interface`** *`type instance`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# interface POS 0/1/0/3`
		Enters interface configuration mode.

Step 10	**`lsp-interval`** *`milliseconds`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# lsp-interval 100`
		(Optional) Configures the amount of time between each LSP sent on an interface.

Step 11	**`csnp-interval`** *`seconds`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# csnp-interval 30 level 1`
		(Optional) Configures the interval at which periodic CSNP packets are sent on broadcast interfaces.
		• Sending more frequent CSNPs means that adjacent routers must work harder to receive them.
		• Sending less frequent CSNP means that differences in the adjacent routers may persist longer.

Step 12	**`retransmit-interval`** *`seconds`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# retransmit-interval 60`
		(Optional) Configures the amount of time that the sending router waits for an acknowledgment before it considers that the LSP was not received and subsequently resends.

Step 13	**`retransmit-throttle-interval`** *`milliseconds`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# retransmit-throttle-interval 1000`
		(Optional) Configures the amount of time between retransmissions on each LSP on a point-to-point interface.
		• This time is usually greater than or equal to the **lsp-interval** command time because the reason for lost LSPs may be that a neighboring router is busy. A longer interval gives the neighbor more time to receive transmissions.

Step 14	**`mesh-group`** `{`*`number`* `|` **`blocked`**`}`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# mesh-group blocked`
		(Optional) Optimizes LSP flooding in NBMA networks with highly meshed, point-to-point topologies.
		• This command is appropriate only for an NBMA network with highly meshed, point-to-point topologies.

	Command or Action	Purpose
Step 15	**end** or **commit** **Example:** RP/0/RP0/CPU0:router(config-isis-if)# end or RP/0/RP0/CPU0:router(config-isis-if)# commit	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: – Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. – Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. – Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 16	**show isis interface** [*type instance* | **level** {**1** | **2**}] [**brief**] **Example:** RP/0/RP0/CPU0:router# show isis interface POS0/1/0/1 brief	(Optional) Displays information about the IS-IS interface.
Step 17	**show isis** [**instance** *instance-id*] **database** [**level** {**1** | **2**}] [**detail** | **summary** | **verbose**] [***** | *lsp-id*] **Example:** RP/0/RP0/CPU0:router# show isis database level 1	(Optional) Displays the IS-IS LSP database.
Step 18	**show isis** [**instance** *instance-id*] **lsp-log** [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router# show isis lsp-log	(Optional) Displays LSP log information.
Step 19	**show isis database-log** [**level** {**1** | **2**}] **Example:** RP/0/RP0/CPU0:router# show isis database-log level 1	(Optional) Display IS-IS database log information.

Configuring Nonstop Forwarding for IS-IS

This task explains how to configure your router with NSF that allows the Cisco IOS XR software to resynchronize the IS-IS link-state database with its IS-IS neighbors after a process restart. The process restart could be due to an:

- RP failover (for a warm restart)
- Simple process restart (due to an IS-IS reload or other administrative request to restart the process)
- IS-IS software upgrade

In all cases, NSF mitigates link flaps and loss of user sessions. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **nsf** {**cisco** | **ietf**}
4. **nsf interface-expires** *number*
5. **nsf interface-timer** *seconds*
6. **nsf lifetime** *seconds*
7. **end**
or
commit
8. **show running-config** [*command*]

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`router isis` *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
		• You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.

Step 3	`nsf {cisco | ietf}`
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# nsf ietf`
		Enables NSF on the next restart.
		• Enter the **cisco** keyword to run IS-IS in heterogeneous networks that might not have adjacent NSF-aware networking devices.
		• Enter the **ietf** keyword to enable IS-IS in homogeneous networks where *all* adjacent networking devices support IETF draft-based restartability.

	Command or Action	Purpose
Step 4	**nsf interface-expires** *number* **Example:** `RP/0/RP0/CPU0:router(config-isis)# nsf interface-expires 1`	Configures the number of resends of an acknowledged NSF-restart acknowledgment. • If the resend limit is reached during the NSF restart, the restart falls back to a cold restart.
Step 5	**nsf interface-timer** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-isis) nsf interface-timer 15`	Configures the number of seconds to wait for each restart acknowledgment.
Step 6	**nsf lifetime** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-isis)# nsf lifetime 20`	Configures the maximum route lifetime following an NSF restart. • This command should be configured to the length of time required to perform a full NSF restart because it is the amount of time that the Routing Information Base (RIB) retains the routes during the restart. • Setting this value too high results in stale routes. • Setting this value too low could result in routes purged too soon.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show running-config** [*command*] **Example:** `RP/0/RP0/CPU0:router# show running-config router isis isp`	(Optional) Displays the entire contents of the currently running configuration file or a subset of that file. • Verify that "nsf" appears in the IS-IS configuration of the NSF-aware device. • This example shows the contents of the configuration file for the "isp" instance only.

Configuring Authentication for IS-IS

This task explains how to configure authentication for IS-IS. This task is optional.

Authentication is available to limit the establishment of adjacencies by using the **hello-password** configuration, and to limit the exchange of LSPs by using the LSP password.

IS-IS supports plain-text authentication, which does not provide security against hackers or other unauthorized users. Plain-text authentication allows you to configure a password to prevent unauthorized networking devices from forming adjacencies with this router. The password is exchanged as plain text and is potentially visible to an agent able to view the IS-IS packets.

IS-IS stores a configured password using simple encryption. However, the plain-text form of the password is used in LSPs, sequence number protocols (SNPs), and hello packets, which would be visible to a process that can view IS-IS packets. The passwords can be entered in plain text (preceded by a 0) or encrypted (preceded by a 7) form.

To set the domain password, configure the **lsp-password** for Level 2; to set the area password, configure the **lsp-password** for Level 1.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **lsp-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**] [**snp send-only**]
4. **interface** *type instance*
5. **hello-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`router isis` *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
		• You can change the level of routing to be performed by a particular routing instance using the **is-type** command.

	Command or Action	Purpose
Step 3	**lsp-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**] [**snp send-only**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# lsp-password hmac-md5 encrypted password1 level 1`	Configures the LSP authentication password. • The **hmac-md5** keyword specifies that the password is used in HMAC-MD5 authentication. • The **text** keyword specifies that the password uses cleartext password authentication. • The **clear** keyword specifies that the password is unencrypted when entered. • The **encrypted** keyword specifies that the password is encrypted using a two-way algorithm when entered. • The **level 1** keyword sets a password for authentication in the area (in Level 1 LSPs and Level SNPs). • The **level 2** keywords set a password for authentication in the backbone (the Level 2 area). • The **send-only** keyword adds authentication to LSP and sequence number protocol data units (SNPs) when they are sent. It does not authenticate received LSPs or SNPs. • The **snp send-only** keyword adds authentication to SNPs when they are sent. It does not authenticate received SNPs. **Note** To disable SNP password checking, the **snp send-only** keywords must be specified in the **lsp-password** command.
Step 4	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface POS 0/1/0/3`	Enters interface configuration mode.

	Command or Action	Purpose
Step 5	**hello-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**] **Example:** `RP/0/RP0/CPU1:router(config-isis-if)# hello-password text clear mypassword`	Configures the authentication password for an IS-IS interface.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring MPLS Traffic Engineering for IS-IS

This task explains how to configure IS-IS for MPLS TE. This task is optional.

For a description of the MPLS TE tasks and commands that allow you to configure the router to support tunnels, configure an MPLS tunnel that IS-IS can use, and troubleshoot MPLS TE, see the *Implementing MPLS Traffic Engineering on Cisco IOS XR Software*.

Prerequisite

Your network must support the following Cisco IOS XR software features before you enable MPLS TE for IS-IS on your router:

- MPLS
- IP Cisco Express Forwarding (CEF)

Note You must enter the commands in the following task list on every IS-IS router in the traffic-engineered portion of your network.

Restrictions

MPLS traffic engineering currently supports only a single IS-IS level and does not support routing and signaling of LSPs over unnumbered IP links. Therefore, do not configure the feature over those links.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** {**ipv4** | **ipv6**} [**unicast**]
4. **mpls traffic-eng level** {**1** | **2**}
5. **mpls traffic-eng router-id** {*ip-address* | *interface-name*}
6. **metric-style wide** [**level** {**1** | **2**}]
7. **end**
   or
   commit
8. **show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel**
9. **show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log**
10. **show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. • You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.
Step 3	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. • This example specifies the unicast IPv6 address family.

Step 4	**mpls traffic-eng level** {**1** | **2**}
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng level 1`
		Configures a router running IS-IS to flood MPLS TE link information into the indicated IS-IS level.

	Command or Action	Purpose

Step 5	**mpls traffic-eng router-id** {*ip-address* | *interface-name*}
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id loopback0`
		Specifies that the MPLS TE router identifier for the node is the IP address and or name associated with a given interface.

Step 6	**metric-style wide** [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1`	Configures a router to generate and accept only wide link metrics in the Level 1 area.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show isis** [**instance** *instance-id*] **mpls traffic-eng tunnel** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng tunnel`	(Optional) Displays MPLS TE tunnel information.
Step 9	**show isis** [**instance** *instance-id*] **mpls traffic-eng adjacency-log** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng adjacency-log`	(Optional) Displays a log of MPLS TE IS-IS adjacency changes.
Step 10	**show isis** [**instance** *instance-id*] **mpls traffic-eng advertisements** **Example:** `RP/0/RP0/CPU0:router# show isis instance isp mpls traffic-eng advertisements`	(Optional) Displays the latest flooded record from MPLS TE.

Tuning Adjacencies for IS-IS on Point-to-Point Interfaces

This task explains how to enable logging of adjacency state changes, alter the timers for IS-IS adjacency packets, and display various aspects of adjacency state. Tuning your IS-IS adjacencies increases network stability when links are congested. This task is optional.

For point-to-point links, IS-IS sends only a single hello for Level 1 and Level 2, which means that the level modifiers are meaningless on point-to-point links. To modify hello parameters for a point-to-point interface, omit the specification of the level options.

The options configurable in the interface submode apply only to that interface. By default, the values are applied to both Level 1 and Level 2.

The **hello-password** command can be used to prevent adjacency formation with unauthorized or undesired routers. This ability is particularly useful on a LAN, where connections to routers with which you have no desire to establish adjacencies are commonly found.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **log adjacency changes**
4. **interface** *type number*
5. **hello-padding** {**disable** | **sometimes**} [**level** {**1** | **2**}]
6. **hello-interval** *seconds* [**level** {**1** | **2**}]
7. **hello-multiplier** *multiplier* [**level** {**1** | **2**}]
8. **hello-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**]
9. **end**
   or
   commit
10. **show isis** [**instance** *instance-id*] **adjacency** [*interface-type interface-instance*] [**detail**] [**systemid** *system-id*]
11. **show isis adjacency-log**
12. **show isis** [**instance** *instance-id*] **interface** [*type instance*] [**brief** | **detail**] [**level** {**1** | **2**}]
13. **show isis** [**instance** *instance-id*] **neighbors** [*interface-type interface-instance*] [**summary**] [**detail**] [**systemid** *system-id*]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router isis`** *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
		• You can change the level of routing to be performed by a particular routing instance using the **is-type** command.

Step 3	**`log adjacency changes`**
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# log adjacency changes`
		Generates a log message when an IS-IS adjacency changes state (up or down).

Step 4	**`interface`** *`type number`*
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# interface POS 0/1/0/3`
		Enters interface configuration mode.

Step 5	**`hello-padding`** `{`**`disable`** `|` **`sometimes`**`} [`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# hello-padding sometimes`
		Configures padding on IS-IS hello PDUs for all IS-IS interfaces on the router.
		• Hello padding applies to only this interface and not to all interfaces.

Step 6	**`hello-interval`** *`seconds`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# hello-interval 6`
		Specifies the length of time between hello packets that the software sends.

Step 7	**`hello-multiplier`** *`multiplier`* `[`**`level`** `{`**`1`** `|` **`2`**`}]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# hello-multiplier 10`
		Specifies the number of IS-IS hello packets a neighbor must miss before the router should declare the adjacency as down.
		• A higher value increases the networks tolerance for dropped packets, but also may increase the amount of time required to detect the failure of an adjacent router.
		• Conversely, not detecting the failure of an adjacent router can result in greater packet loss.

	Command or Action	Purpose
Step 8	**hello-password** {**hmac-md5** | **text**} {**clear** | **encrypted**} *password* [**level** {**1** | **2**}] [**send-only**] **Example:** `RP/0/RP0/CPU1:router(config-isis-if)# hello-password text clear mypassword`	Specifies that this system include authentication in the hello packets and requires successful authentication of the hello packet from the neighbor to establish an adjacency.

Step 9	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show isis** [**instance** *instance-id*] **adjacency** [*interface-type interface-instance*] [**detail**] [**systemid** *system-id*] **Example:** `RP/0/RP0/CPU0:router# show isis instance isp adjacency ipv4`	(Optional) Displays IS-IS adjacencies.
Step 11	**show isis adjacency-log** **Example:** `RP/0/RP0/CPU1:router# show isis adjacency-log`	(Optional) Displays a log of the most recent adjacency state transitions.

	Command or Action	Purpose
Step 12	**show isis** [**instance** *instance-id*] **interface** [*type instance*] [**brief** | **detail**] [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router# show isis interface POS 0/1/0/1 brief`	(Optional) Displays information about the IS-IS interface.
Step 13	**show isis** [**instance** *instance-id*] **neighbors** [*interface-type interface-instance*] [**summary**] [**detail**] [**systemid** *system-id*] **Example:** `RP/0/RP0/CPU0:router# show isis neighbors summary`	(Optional) Displays information about IS-IS neighbors.

Setting SPF Interval for a Single-Topology IPv4 and IPv6 Configuration

This task explains how to make adjustments to the SPF calculation to tune router performance. This task is optional.

Because the SPF calculation computes routes for a particular topology, the tuning attributes are located in the router address family configuration submode. SPF calculation computes routes for Level 1 and Level 2 separately.

When IPv4 and IPv6 address families are used in a single-topology mode, only a single SPF for the IPv4 topology exists. The IPv6 topology “borrows” the IPv4 topology; therefore, no SPF calculation is required for IPv6. To tune the SPF calculation parameters for single-topology mode, configure the **address-family ipv4 unicast** command.

The incremental SPF algorithm can be enabled separately. When enabled, the incremental shortest path first (ISPF) is not employed immediately. Instead, the full SPF algorithm is used to “seed” the state information required for the ISPF to run. The startup delay prevents the ISPF from running for a specified interval after an IS-IS restart (to permit the database to stabilize). After the startup delay elapses, the ISPF is principally responsible for performing all of the SPF calculations. The reseed interval enables a periodic running of the full SPF to ensure that the iSFP state remains synchronized.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** {**ipv4** | **ipv6**} [**unicast**]
4. **spf-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...}[**level** {**1** | **2**}]
5. **ispf [startup-delay** *seconds*] [**level** {**1** | **2**}]
6. **ispf startup-delay** *seconds* [**level** {**1** | **2**}]
7. **end**
   or
   commit
8. **show isis** [**instance** *instance-id*] **spf-log** [**level** {**1** | **2**}] [**ipv4** | **ipv6**] [**unicast**] [**ispf** | **fspf** | **prc**] [**detail**] [**internal**] [**last** *number* | **first** *number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. - You can change the level of routing to be performed by a particular routing instance using the **is-type** router configuration command.
Step 3	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. - This example specifies the unicast IPv6 address family.
Step 4	**spf-interval** {[**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] ...} [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# spf-interval initial-wait 10 maximum-wait 30`	(Optional) Controls the minimum time between successive SPF calculations. - This value imposes a delay in the SPF computation after an event trigger and enforces a minimum elapsed time between SPF runs. - If this value is configured too low, the router can lose too many CPU resources when the network is unstable. - Configuring the value too high delays changes in the network topology that result in lost packets. - The SPF interval does not apply to the running of the ISPF because that algorithm runs immediately on receiving a changed LSP.
Step 5	**ispf [startup-delay** *seconds*] [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# ispf`	(Optional) Configures incremental IS-IS ISPF to calculate network topology.
Step 6	**ispf startup-delay** *seconds* [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# ispf startup-delay 600`	(Optional) Configures the time delay between the starting of the IS-IS instance and the activation of ISPF.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show isis** [**instance** *instance-id*] **spf-log** [**level** {**1** | **2**}] [**ipv4** | **ipv6**] [**unicast**] [**ispf** | **fspf** | **prc**] [**detail**] [**internal**] [**last** *number* | **first** *number*] **Example:** `RP/0/RP0/CPU0:router# show isis instance 1 spf-log ipv4`	(Optional) Displays how often and why the router has run a full SPF calculation.

Enabling Multicast-Intact for IS-IS

This optional task describes how to enable multicast-intact for IS-IS routes that use IPv4 addresses.

Summary Steps

1. **configure**
2. **router isis** *instance-id*
3. **address-family** {**ipv4** | **ipv6**} [**unicast**]
4. **mpls traffic-eng multicast-intact**
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router isis`** *`instance-id`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router isis isp`
		Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. In this example, the IS-IS instance is called isp.

Step 3	**`address-family`** `{`**`ipv4`** `|` **`ipv6`**`} [`**`unicast`**`]`
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# address-family ipv4`
		Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. This example specifies the unicast IPv4 address family.

Step 4	**`mpls traffic-eng multicast-intact`**
	Example:
	`RP/0/RP0/CPU0:router(config-isis)# mpls traffic-eng multicast-intact`
		Enables multicast-intact.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Customizing Routes for IS-IS

This task describes how to perform route functions that include injecting default routes into your IS-IS routing domain and redistributing routes learned at one IS-IS level into a different level. This task is optional.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **set-overload-bit** [**on-startup** {*delay* | **wait-for-bgp**}] [**level** {**1** | **2**}]
4. **address-family** {**ipv4** | **ipv6**} [**unicast**]
5. **default-information originate** [**route-map** *map-name*]
6. **redistribute isis** *instance* [**level-1** | **level-2** | **level-1-2**] [**metric** *metric*] [**metric-type** {**internal** | **external**}] **policy** *policy-name*]
7. **summary-prefix** [*address*/*prefix-length*] [**level** {**1** | **2**}]
   or
   summary-prefix [*ipv6-prefix*/*prefix-length*] [**level** {**1** | **2**}]
8. **maximum-paths** *route-number*
9. **distance** *weight* [*address/prefix-length* [*route-list-name*]]
10. **set-attached-bit**
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing process, and places the router in router configuration mode. • By default, all IS-IS instances are automatically Level 1 and Level 2. You can change the level of routing to be performed by a particular routing instance using the **is-type** command.
Step 3	**set-overload-bit** [**on-startup** {*delay* | **wait-for-bgp**}] [**level** {**1** | **2**}] **Example:** `RP/0/RP0/CPU0:router(config-isis)# set-overload-bit`	(Optional) Sets the overload bit. **Note** The configured overload bit behavior does not apply to NSF restarts because the NSF restart does not set the overload bit during restart.
Step 4	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv6 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. • This example specifies the unicast IPv6 address family.

	Command or Action	Purpose
Step 5	**default-information originate** [**route-map** *map-name*] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# default-information originate`	(Optional) Injects a default IPv4 or IPv6 route into an IS-IS routing domain. • The **route-map** keyword and *map-name* argument specify the conditions under which the IPv4 or IPv6 default route is advertised. • If the **route-map** keyword is omitted, then the IPv4 or IPv6 default route is unconditionally advertised at Level 2.
Step 6	**redistribute isis** *instance* [**level-1** | **level-2** | **level-1-2**] [**metric** *metric*] [**metric-type** {**internal** | **external**}] [**policy** *policy-name*] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# redistribute isis 2 level-1`	(Optional) Redistributes routes from one IS-IS instance into another instance. • In this example, an IS-IS instance redistributes IS-IS instance 2 routes into its Level 1 area.

Step 7	**summary-prefix** [*address***/***prefix-length*] [**level** {**1** | **2**}]
	or
	summary-prefix [*ipv6-prefix***/***prefix-length*] [**level** {**1** | **2**}]
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 10.1.0.0/16 level 1`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# summary-prefix 3003:xxxx::/24 level 1`
		(Optional) Allows a Level 1-2 router to summarize Level 1 IPv4 and IPv6 prefixes at Level 2, instead of advertising the Level 1 prefixes directly when the router advertises the summary.
		• This example specifies an IPv4 address and mask.
		or
		• This example specifies an IPv6 prefix, and the command must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.
		• Note that IPv6 prefixes must be configured only in the IPv6 router address family configuration submode, and IPv4 prefixes in the IPv4 router address family configuration submode.

Step 8	**maximum-paths** *route-number* **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# maximum-paths 16`	(Optional) Configures the maximum number of parallel paths allowed in a routing table.
Step 9	**distance** *weight* [*address*/*prefix-length* [*route-list-name*]] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# distance 90`	(Optional) Defines the administrative distance assigned to routes discovered by the IS-IS protocol. • A different administrative distance may be applied for IPv4 and IPv6.

	Command or Action	Purpose
Step 10	**set-attached-bit** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# set-attached-bit`	(Optional) Configures an IS-IS instance with an attached bit in the Level 1 LSP.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing IS-IS on Cisco IOS XR Software

This section provides the following configuration examples:

- Configuring Single-Topology IS-IS for IPv6: Example, page RC-122
- Configuring Multitopology IS-IS for IPv6: Example, page RC-123
- Redistributing IS-IS Routes Between Multiple Instances: Example, page RC-123

Configuring Single-Topology IS-IS for IPv6: Example

The following example shows single-topology mode being enabled, an IS-IS instance being created, the NET being defined, IPv6 being configured along with IPv4 on an interface, and IPv4 link topology being used for IPv6.

This configuration allows POS interface 0/3/0/0 to form adjacencies for both IPv4 and IPv6 addresses.

```
router isis isp
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface POS0/3/0/0
```

```
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
  !
  exit
!
interface POS0/3/0/0
 ipv4 address 10.0.1.3 255.255.255.0
 ipv6 address 2001::1/64
```

Configuring Multitopology IS-IS for IPv6: Example

The following example shows multitopology IS-IS being configured in IPv6. You need not enable IS-IS for IPv6 globally on the router.

```
router isis isp
 net 49.0000.0000.0001.00
 interface POS0/3/0/0
  address-family ipv6 unicast
  metric-style wide level 1
  exit
!
interface POS0/3/0/0
 ipv6 address 2001::1/64
```

Redistributing IS-IS Routes Between Multiple Instances: Example

The following example shows the attached bit being set for a Level 1 instance. This example shows the other Level 1 routers in the area being informed that this router is a suitable candidate to get from the area to the backbone. The Level 1 instance is also propagating routes to the Level 2 instance using redistribution. Note that the administrative distance is explicitly configured higher on the Level 2 instance to ensure that Level 1 routes are preferred.

```
router isis 1
  is-type level-2-only
 net 49.0001.0001.0001.0001.00
 address-family ipv4 unicast
  distance 116
  redistribute isis 2 level 2
!
interface POS0/3/0/0
 address-family ipv4 unicast
!
!
router isis 2
 is-type level-1
 net 49.0002.0001.0001.0002.00
 address-family ipv4 unicast
  set-attached-bit
!
interface POS0/1/0/0
 address-family ipv4 unicast
```

Where to Go Next

To implement more IP routing protocols, see the following document modules:

- *Implementing OSPF on Cisco IOS XR Software*
- *Implementing BGP on Cisco IOS XR Software*

Additional References

The following sections provide references related to implementing IS-IS on Cisco IOS XR software.

Related Documents

Related Topic	Document Title
IS-IS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*, Release 3.2
MPLS TE feature information	*Implementing MPLS Traffic Engineering on Cisco IOS XR Software* module in the *Cisco IOS XR Multiprotocol Label Switching Configuration Guide*, Release 3.2

Standards

Standards	Title
Draft-ietf-isis-ipv6-05.txt	*Routing IPv6 with IS-IS*, by Christian E. Hopps
Draft-ietf-isis-wg-multi-topology-06.txt	*M-ISIS: Multi Topology (MT) Routing in IS-IS,* by Tony Przygienda, Naiming Shen, and Nischal Sheth
Draft-ietf-isis-traffic-05.txt	*IS-IS Extensions for Traffic Engineering*, by Henk Smit and Toni Li
Draft-ietf-isis-restart-04.txt	*Restart Signalling for IS-IS*, by M. Shand and Les Ginsberg
Draft-ietf-isis-igp-p2p-over-lan-05.txt	*Point-to-point operation over LAN in link-state routing protocols*, by Naiming Shen

MIBs

MIBs	MIBs Link
There are no applicable MIBs for this module.	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1142	*OSI IS-IS Intra-domain Routing Protocol*
RFC 1195	*Use of OSI IS-IS for Routing in TCP/IP and Dual Environments*
RFC 2763	*Dynamic Hostname Exchange Mechanism for IS-IS*
RFC 2966	*Domain-wide Prefix Distribution with Two-Level IS-IS*
RFC 2973	*IS-IS Mesh Groups*
RFC 3277	*IS-IS Transient Blackhole Avoidance*
RFC 3373	*Three-Way Handshake for IS-IS Point-to-Point Adjacencies*
RFC 3567	*IS-IS Cryptographic Authentication*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing OSPF on Cisco IOS XR Software

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets.

Implementing OSPF version 3 (OSPFv3) expands on OSPF Version 2, to provide support for IPv6 routing prefixes.

This module describes the concepts and tasks you need to implement both versions of OSPF on your Cisco IOS XR router. The term "OSPF" implies both versions of the routing protocol, unless otherwise noted.

Note For more information about OSPF on the Cisco IOS XR software and complete descriptions of the OSPF commands listed in this module, see the "Related Documents" section of this module. To locate documentation for other commands that might appear during execution of a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing OSPF on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.
Release 3.2.2	Support was added for the multicast-intact feature.

Contents

Prerequisites for Implementing OSPF on Cisco IOS XR Software

The following are prerequisites for implementing OSPF on Cisco IOS XR Software:

- You must be in a user group associated with a task group that includes the proper task IDs for OSPF commands. Task IDs for commands are listed in the Cisco IOS XR Task ID Reference Guide. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- Configuration tasks for OSPFv3 assume that you are familiar with IPv6 addressing and basic configuration. See the *Implementing Network Stack IPv4 and IPv6 on Cisco IOS XR Software* module of the *Cisco IOS XR IP Addresses and Services Configuration Guide* for information on IPv6 routing and addressing.
- Before you enable OSPFv3 on an interface, you must perform the following tasks:
  - Complete the OSPF network strategy and planning for your IPv6 network. For example, you must decide whether multiple areas are required.
  - Enable IPv6 on the interface.
- Configuring authentication (IP Security) is an optional task. If you choose to configure authentication, you must first decide whether to configure plain text or Message Digest 5 (MD5) authentication, and whether the authentication applies to an entire area or specific interfaces.

Information About Implementing OSPF on Cisco IOS XR Software

To implement OSPF you need to understand the following concepts:

- Nonstop Forwarding for OSPF Version 2, page RC-140
- Load Balancing in OSPF Version 2 and OSPFv3, page RC-141

OSPF Functional Overview

OSPF is a routing protocol for IP. It is a link-state protocol, as opposed to a distance-vector protocol. A link-state protocol makes its routing decisions based on the states of the links that connect source and destination machines. The state of the link is a description of that interface and its relationship to its neighboring networking devices. The interface information includes the IP address of the interface, network mask, type of network to which it is connected, routers connected to that network, and so on. This information is propagated in various types of link-state advertisements (LSAs).

A router stores the collection of received link-state advertisement (LSA) data in a link-state database. This database includes LSA data for the links of the router. The contents of the database, when subjected to the Dijkstra algorithm, extract data to create an OSPF routing table. The difference between the database and the routing table is that the database contains a complete collection of raw data; the routing table contains a list of shortest paths to known destinations through specific router interface ports.

OSPF is the IGP of choice because it scales to large networks. It uses areas to partition the network into more manageable sizes and to introduce hierarchy in the network. A router is attached to one or more areas in a network. All of the networking devices in an area maintain the same complete database information about the link states in their area only. They do not know about all link states in the network. The agreement of the database information among the routers in the area is called convergence.

At the intradomain level, OSPF can import routes learned using Intermediate System-to-Intermediate System (IS-IS). OSPF routes can also be exported into IS-IS. At the interdomain level, OSPF can import routes learned using Border Gateway Protocol (BGP). OSPF routes can be exported into BGP.

Unlike Routing Information Protocol (RIP), OSPF does not provide periodic routing updates. On becoming neighbors, OSPF routers establish an adjacency by exchanging and synchronizing their databases. After that, only changed routing information is propagated. Every router in an area advertises the costs and states of its links, sending this information in an LSA. This state information is sent to all OSPF neighbors one hop away. All the OSPF neighbors, in turn, send the state information unchanged. This flooding process continues until all devices in the area have the same link-state database.

To determine the best route to a destination, the software sums all of the costs of the links in a route to a destination. After each router has received routing information from the other networking devices, it runs the shortest path first (SPF) algorithm to calculate the best path to each destination network in the database.

The networking devices running OSPF detect topological changes in the network, flood link-state updates to neighbors, and quickly converge on a new view of the topology. Each OSPF router in the network soon has the same topological view again. OSPF allows multiple equal-cost paths to the same destination. Since all link-state information is flooded and used in the SPF calculation, multiple equal cost paths can be computed and used for routing.

On broadcast and nonbroadcast multiaccess (NBMA) networks, the designated router (DR) or backup DR performs the LSA flooding. On point-to-point networks, flooding simply exits an interface directly to a neighbor.

OSPF runs directly on top of IP; it does not use TCP or User Datagram Protocol (UDP). OSPF performs its own error correction by means of checksums in its packet header and LSAs.

In OSPFv3, the fundamental concepts are the same as OSPF Version 2, except that support is added for the increased address size of IPv6. New LSA types are created to carry IPv6 addresses and prefixes, and the protocol runs on an individual link basis rather than on an individual IP-subnet basis.

OSPF typically requires coordination among many internal routers: Area Border Routers (ABRs), which are routers attached to multiple areas, and Autonomous System Border Routers (ASBRs) that export reroutes from other sources (for example, IS-IS, BGP, or static routes) into the OSPF topology. At a minimum, OSPF-based routers or access servers can be configured with all default parameter values, no authentication, and interfaces assigned to areas. If you intend to customize your environment, you must ensure coordinated configurations of all routers.

Key Features Supported in the Cisco IOS XR OSPF Implementation

The Cisco IOS XR implementation of OSPF conforms to the OSPF Version 2 and OSPF Version 3 specifications detailed in the Internet RFC 2328 and RFC 2740, respectively.

The following key features are supported in the Cisco IOS XR implementation:

- Hierarchy—CLI hierarchy is supported.
- Inheritance—CLI inheritance is supported.
- Stub areas—Definition of stub areas is supported.
- NSF—Nonstop forwarding is supported.
- SPF throttling—Shortest path first throttling feature is supported.
- LSA throttling—LSA throttling feature is supported.
- Fast convergence—SPF and LSA throttle timers are set, configuring fast convergence. The OSPF LSA throttling feature provides a dynamic mechanism to slow down LSA updates in OSPF during network instability. LSA throttling also allows faster OSPF convergence by providing LSA rate limiting in milliseconds.
- Route redistribution—Routes learned using any IP routing protocol can be redistributed into any other IP routing protocol.
- Authentication—Plain text and MD5 authentication among neighboring routers within an area is supported.
- Routing interface parameters—Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, router “dead” and hello intervals, and authentication key.
- Virtual links—Virtual links are supported.
- Not-so-stubby area (NSSA)—RFC 1587 is supported.
- OSPF over demand circuit—RFC 1793 is supported.

Comparison of Cisco IOS XR OSPFv3 and OSPFv2

Much of the OSPFv3 protocol is the same as in OSPFv2. OSPFv3 is described in RFC 2740.

The key differences between the Cisco IOS XR OSPFv3 and OSPFv2 protocols are as follows:

- OSPFv3 expands on OSPFv2 to provide support for IPv6 routing prefixes and the larger size of IPv6 addresses.
- When using an NBMA interface in OSPFv3, users must manually configure the router with the list of neighbors. Neighboring routers are identified by the link local address of the attached interface of the neighbor.
- Unlike in OSPFv2, multiple OSPFv3 processes can be run on a link.
- LSAs in OSPFv3 are expressed as “prefix and prefix length” instead of “address and mask.”
- The router ID is a 32-bit number with no relationship to an IPv6 address.

Importing Addresses into OSPFv3

When importing into OSPFv3 the set of addresses configured on an OSPFv3 interface, users cannot select specific addresses to be imported. Either all addresses are imported or no addresses are imported.

OSPF Hierarchical CLI and CLI Inheritance

Cisco IOS XR software introduces new OSPF configuration fundamentals consisting of hierarchical CLI and CLI inheritance.

Hierarchical CLI is the grouping of related network component information at defined hierarchical levels such as at the router, area, and interface levels. Hierarchical CLI allows for easier configuration, maintenance, and troubleshooting of OSPF configurations. When configuration commands are displayed together in their hierarchical context, visual inspections are simplified. Hierarchical CLI is intrinsic for CLI inheritance to be supported.

With CLI inheritance support, you need not explicitly configure a parameter for an area or interface. In Cisco IOS XR, the parameters of interfaces in the same area can be exclusively configured with a single command, or parameter values can be inherited from a higher hierarchical level—such as from the area configuration level or the router ospf configuration levels.

For example, the hello interval value for an interface is determined by this precedence “IF” statement:

> If the **hello interval** command is configured at the interface configuration level, then use the interface configured value, else
>
> If the **hello interval** command is configured at the area configuration level, then use the area configured value, else
>
> If the **hello interval** command is configured at the router ospf configuration level, then use the router ospf configured value, else
>
> Use the default value of the command.

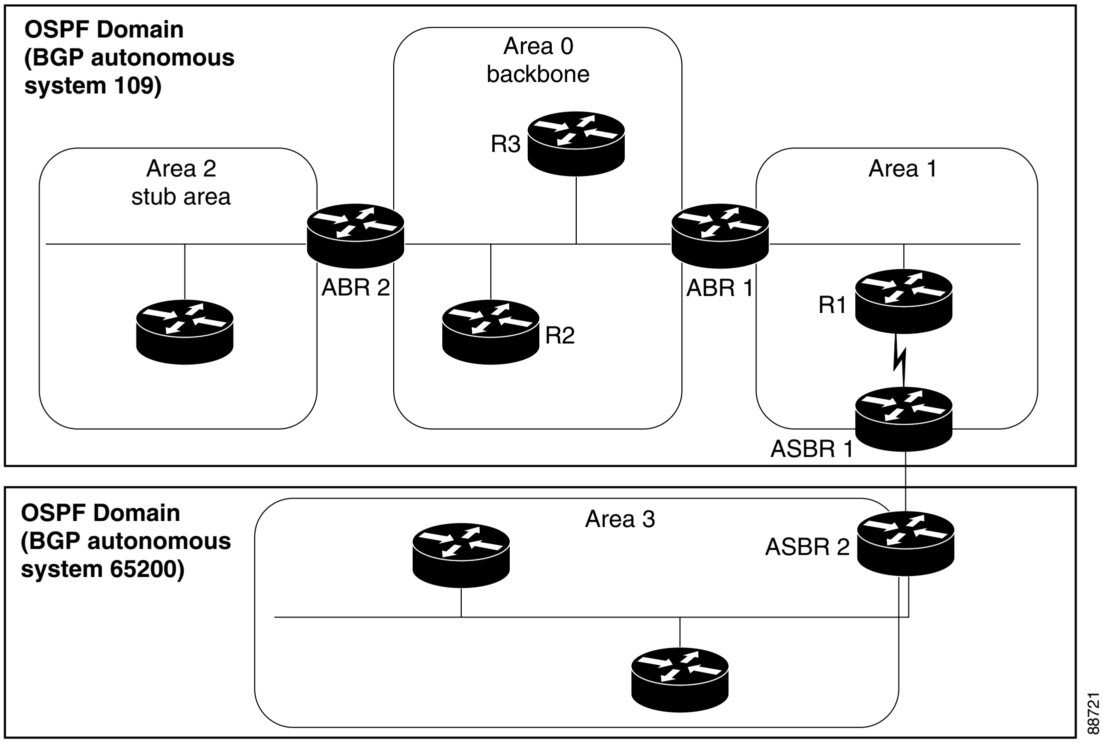
Tip Understanding hierarchical CLI and CLI inheritance saves you considerable configuration time. See the "Configuring Authentication at Different Hierarchical Levels for OSPF Version 2" section on page RC-155 to understand how to implement these fundamentals. In addition, Cisco IOS XR examples are provided in the "Configuration Examples for Implementing OSPF on Cisco IOS XR Software" section on page RC-187.

OSPF Routing Components

Before implementing OSPF, you must know what the routing components are and what purpose they serve. They consist of the autonomous system, area types, interior routers, ABRs, and ASBRs.

Figure 6 illustrates the routing components in an OSPF network topology.

Figure 6 OSPF Routing Components



Autonomous Systems

The autonomous system is a collection of networks, under the same administrative control, that share routing information with each other. An autonomous system is also referred to as a routing domain. Figure 6 shows two autonomous systems: A and B. An autonomous system can consist of one or more OSPF areas.

Areas

Areas allow the subdivision of an autonomous system into smaller, more manageable networks or sets of adjacent networks. As shown in Figure 6, autonomous system A consists of three areas: Area 0, Area 1, and Area 2.

OSPF hides the topology of an area from the rest of the autonomous system. The network topology for an area is visible only to routers inside that area. When OSPF routing is within an area, it is called *intra-area routing*. This routing limits the amount of link-state information flood into the network, reducing routing traffic. It also reduces the size of the topology information in each router, conserving processing and memory requirements in each router.

Also, the routers within an area cannot see the detailed network topology outside the area. Because of this restricted view of topological information, you can control traffic flow between areas and reduce routing traffic when the entire autonomous system is a single routing domain.

Backbone Area

A backbone area is responsible for distributing routing information between multiple areas of an autonomous system. OSPF routing occurring outside of an area is called *interarea routing*.

The backbone itself has all properties of an area. It consists of ABRs, routers, and networks only on the backbone. As shown in Figure 6, Area 0 is an OSPF backbone area. Any OSPF backbone area has a reserved area ID of 0.0.0.0.

Stub Area

A stub area is an area that does not accept or detailed network information external to the area. A stub area typically has only one router that interfaces the area to the rest of the autonomous system. The stub ABR advertises a single default route to external destinations into the stub area. Routers within a stub area use this route for destinations outside the area and the autonomous system. This relationship conserves LSA database space that would otherwise be used to store external LSAs flooded into the area. In Figure 6, Area 2 is a stub area that is reached only through ABR 2. Area 0 cannot be a stub area.

Not-so-Stubby Area (NSSA)

NSSA is similar to the stub area. NSSA does not flood Type 5 external LSAs from the core into the area, but can import autonomous system external routes in a limited fashion within the area.

NSSA allows importing of Type 7 autonomous system external routes within an NSSA area by redistribution. These Type 7 LSAs are translated into Type 5 LSAs by NSSA ABRs, which are flooded throughout the whole routing domain. Summarization and filtering are supported during the translation.

Use NSSA to simplify administration if you are a network administrator that must connect a central site using OSPF to a remote site that is using a different routing protocol.

Before NSSA, the connection between the corporate site border router and remote router could not be run as an OSPF stub area because routes for the remote site could not be redistributed into a stub area, and two routing protocols needed to be maintained. A simple protocol like RIP was usually run and handled the redistribution. With NSSA, you can extend OSPF to cover the remote connection by defining the area between the corporate router and remote router as an NSSA. Area 0 cannot be an NSSA.

Routers

The OSPF network is composed of ABRs, ASBRs, and interior routers.

Area Border Routers (ABR)

ABRs are routers with multiple interfaces that connect directly to networks in two or more areas. An ABR runs a separate copy of the OSPF algorithm and maintains separate routing data for each area that is attached to, including the backbone area. ABRs also send configuration summaries for their attached areas to the backbone area, which then distributes this information to other OSPF areas in the autonomous system. In Figure 6, there are two ABRs. ABR 1 interfaces Area 1 to the backbone area. ABR 2 interfaces the backbone Area 0 to Area 2, a stub area.

Autonomous System Boundary Routers (ASBR)

ASBRs provide connectivity from one autonomous system to another system. ASBRs exchange their autonomous system routing information with boundary routers in other autonomous systems. Every router inside an autonomous system knows how to reach the boundary routers for its autonomous system.

ASBRs can import external routing information from other protocols like BGP and redistribute them as AS-external (ASE) Type 5 LSAs to the OSPF network. If the Cisco IOS XR router is an ASBR, you can configure it to advertise VIP addresses for content as autonomous system external routes. In this way, ASBRs flood information about external networks to routers within the OSPF network.

ASBR routes can be advertised as a Type 1 or Type 2 ASE. The difference between Type 1 and Type 2 is how the cost is calculated. For a Type 2 ASE, only the external cost (metric) is considered when multiple paths to the same destination are compared. For a Type 1 ASE, the combination of the external cost and cost to reach the ASBR is used. Type 2 external cost is the default and is always more costly than an OSPF route and used only if no OSPF route exists.

Interior Routers

The interior routers (such as R1 in Figure 6) attached to one area (for example, all the interfaces reside in the same area).

OSPF Process and Router ID

An OSPF process is a logical routing entity running OSPF in a physical router. This logical routing entity should not be confused with the logical routing feature that allows a system administrator (known as the Cisco IOS XR Owner) to partition the physical box into separate routers.

A physical router can run multiple OSPF processes, although the only reason to do so would be to connect two or more OSPF domains. Each process has its own link-state database. The routes in the routing table are calculated from the link-state database. One OSPF process does not share routes with another OSPF process unless the routes are redistributed.

Each OSPF process is identified by a router ID. The router ID must be unique across the entire routing domain. OSPFv2 obtains a router ID from the following sources, in order of decreasing preference:

OSPF attempts to obtain a router ID in the following ways (in order of preference):

- The 32-bit numeric value specified by the OSPF **router-id** command in router configuration mode. (This value can be any 32-bit value. It is not restricted to the IPv4 addresses assigned to interfaces on this router, and need not be a routable IPv4 address.)

- The primary IPv4 address of the interface specified by the OSPF **router-id** command.
- The 32-bit numeric value specified by the **router-id** command in global configuration mode. (This value must be an IPv4 address assigned to an interface on this router.)
- By using the highest IPv4 address on a loopback interface in the system if the router is booted with saved loopback address configuration.
- The primary IPv4 address of an interface over which this OSPF process is running.

We recommend that the router ID be set by the **router-id** command in router configuration mode. Separate OSPF processes could share the same router ID, in which case they cannot reside in the same OSPF routing domain.

Supported OSPF Network Types

OSPF classifies different media into the following three types of networks by default:

- NBMA networks (POS)
- Point-to-point networks (POS)
- Broadcast networks (Gigabit Ethernet)

You can configure your Cisco IOS XR network as either a broadcast or an NBMA network. Using this feature, you can configure broadcast networks as NBMA networks when, for example, you have routers in your network that do not support multicast addressing.

Route Authentication Methods for OSPF Version 2

OSPF Version 2 supports two types of route authentication: plain text authentication and MD5 authentication. By default, no authentication is enabled (referred to as null authentication in RFC 2178).

Both plain text and MD5 authentication are performed on changed routing information that arrive on an interface. The sender and receiver must know the authentication password or key. For both types of authentication, a router sends a routing update packet with a key and corresponding key number. The receiving router checks the key number and key against its own stored key number and key. If the key numbers and keys match, the router accepts the routing update packet. If they do not match, the routing update is discarded.

Plain Text Authentication

Plain text authentication (also known as Type 1 authentication) uses a password that travels on the physical medium and is easily visible to someone that does not have access permission and could use the password to infiltrate a network. Therefore, plain text authentication does not provide security. It might protect against a faulty implementation of OSPF or a misconfigured OSPF interface trying to send erroneous OSPF packets.

MD5 Authentication

MD5 authentication provides a means of security. No password travels on the physical medium. Instead, the router uses MD5 to produce a message digest of the OSPF packet plus the key, which is sent on the physical medium. Using MD5 authentication prevents a router from accepting unauthorized or deliberately malicious routing updates, which could compromise your network security by diverting your traffic.

Note MD5 authentication supports multiple keys, requiring that a key number be associated with a key.

Authentication Strategies

Authentication can be specified for an entire process or area, or on an interface or a virtual link. An interface or virtual link can be configured for only one type of authentication, not both. Authentication configured for an interface or virtual link overrides authentication configured for the area or process.

If you intend for all interfaces in an area to use the same type of authentication, you can configure fewer commands if you use the **area authentication** command (and specify the **message-digest** keyword if you want the entire area to use MD5 authentication). This strategy requires fewer commands than specifying authentication for each interface.

Key Rollover

To support the changing of a plain text key or MD5 key in an operational network without disrupting OSPF adjacencies (and hence the topology), a key rollover mechanism is supported. As a network administrator configures the new key into the multiple networking devices that communicate, some time exists when different devices are using both a new key and an old key. If an interface is configured with a new key, the software sends two copies of the same packet, each authenticated by the old key and new key. The software tracks which devices start using the new key, and the software stops sending duplicate packets after it detects that all of its neighbors are using the new key. The software then discards the old key. The network administrator must then remove the old key from each the configuration file of each router.

Neighbors and Adjacency for OSPF

Routers that share a segment (Layer 2 link between two interfaces) become neighbors on that segment. OSPF uses the hello protocol as a neighbor discovery and keep alive mechanism. The hello protocol involves receiving and periodically sending hello packets out each interface. The hello packets list all known OSPF neighbors on the interface. Routers become neighbors when they see themselves listed in the hello packet of the neighbor. After two routers are neighbors, they may proceed to exchange and synchronize their databases, which creates an adjacency. On broadcast and NBMA networks all neighboring routers have an adjacency.

Designated Router (DR) for OSPF

On point-to-point and point-to-multipoint networks, the Cisco IOS XR software floods routing updates to immediate neighbors. No DR or backup DR (BDR) exists; all routing information is flooded to each router.

On broadcast or NBMA segments only, OSPF minimizes the amount of information being exchanged on a segment by choosing one router to be a DR and one router to be a BDR. Thus, the routers on the segment have a central point of contact for information exchange. Instead of each router exchanging routing updates with every other router on the segment, each router exchanges information with the DR and BDR. The DR and BDR relay the information to the other routers. On broadcast network segments the number of OSPF packets is further reduced by the DR and BDR sending such OSPF updates to a multicast IP address that all OSPF routers on the network segment are listening on.

The software looks at the priority of the routers on the segment to determine which routers are the DR and BDR. The router with the highest priority is elected the DR. If there is a tie, then the router with the higher router ID takes precedence. After the DR is elected, the BDR is elected the same way. A router with a router priority set to zero is ineligible to become the DR or BDR.

Default Route for OSPF

Type 5 (ASE) LSAs are generated and flooded to all areas except stub areas. For the routers in a stub area to be able to route packets to destinations outside the stub area, a default route is injected by the ABR attached to the stub area.

The cost of the default route is 1 (default) or is determined by the value specified in the **default-cost** command.

Link-State Advertisement Types for OSPF Version 2

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the links that the router has within a single area, and the cost of each link. These LSAs are flooded within an area only. The LSA indicates if the router can compute paths based on quality of service (QoS), whether it is an ABR or ASBR, and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks.
- Network LSA (Type 2)—Describes the link state and cost information for all routers attached a multiaccess network segment. This LSA lists all the routers that have interfaces attached to the network segment. It is the job of the designated router of a network segment to generate and track the contents of this LSA.
- Summary LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or a set of networks aggregated into one prefix. Only ABRs generate summary LSAs.
- Summary LSA for ASBRs (Type 4)—Advertises and ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF.

Link-State Advertisement Types for OSPFv3

Each of the following LSA types has a different purpose:

- Router LSA (Type 1)—Describes the link state and costs of a the router link to the area. These LSAs are flooded within an area only. The LSA indicates whether the router is an ABR or ASBR and if it is one end of a virtual link. Type 1 LSAs are also used to advertise stub networks. In OSPFv3, these LSAs have no address information and are network protocol independent. In OSPFv3, router interface information may be spread across multiple router LSAs. Receivers must concatenate all router LSAs originated by a given router before running the SPF calculation.

- Network LSA (Type 2)—Describes the link state and cost information for all routers attached to a multiaccess network segment. This LSA lists all OSPF routers that have interfaces attached to the network segment. Only the elected designated router for the network segment can generate and track the network LSA for the segment. In OSPFv3, network LSAs have no address information and are network-protocol-independent.
- Interarea-prefix LSA for ABRs (Type 3)—Advertises internal networks to routers in other areas (interarea routes). Type 3 LSAs may represent a single network or set of networks aggregated into one prefix. Only ABRs generate Type 3 LSAs. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.
- Interarea-router LSA for ASBRs (Type 4)—Advertises an ASBR and the cost to reach it. Routers that are trying to reach an external network use these advertisements to determine the best path to the next hop. ABRs generate Type 4 LSAs.
- Autonomous system external LSA (Type 5)—Redistributes routes from another autonomous system, usually from a different routing protocol into OSPF. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.
- Link LSA (Type 8)—Has link-local flooding scope and is never flooded beyond the link with which it is associated. Link LSAs provide the link-local address of the router to all other routers attached to the link or network segment, inform other routers attached to the link of a list of IPv6 prefixes to associate with the link, and allow the router to assert a collection of Options bits to associate with the network LSA that is originated for the link.
- Intra-area-prefix LSAs (Type 9)—A router can originate multiple intra-area-prefix LSAs for every router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or network LSA and contains prefixes for stub and transit networks.

An address prefix occurs in almost all newly defined LSAs. The prefix is represented by three fields: Prefix Length, Prefix Options, and Address Prefix. In OSPFv3, addresses for these LSAs are expressed as "prefix and prefix length" instead of "address and mask." The default route is expressed as a prefix with length 0.

Inter-area-prefix and intra-area-prefix LSAs carry all IPv6 prefix information that, in IPv4, is included in router LSAs and network LSAs. The Options field in certain LSAs (router LSAs, network LSAs, interarea-router LSAs, and link LSAs) has been expanded to 24 bits to provide support for OSPF in IPv6.

In OSPFv3, the sole function of link-state ID in interarea-prefix LSAs, interarea-router LSAs, and autonomous system external LSAs is to identify individual pieces of the link-state database. All addresses or router IDs that are expressed by the link-state ID in OSPF Version 2 are carried in the body of the LSA in OSPFv3.

Virtual Link and Transit Area for OSPF

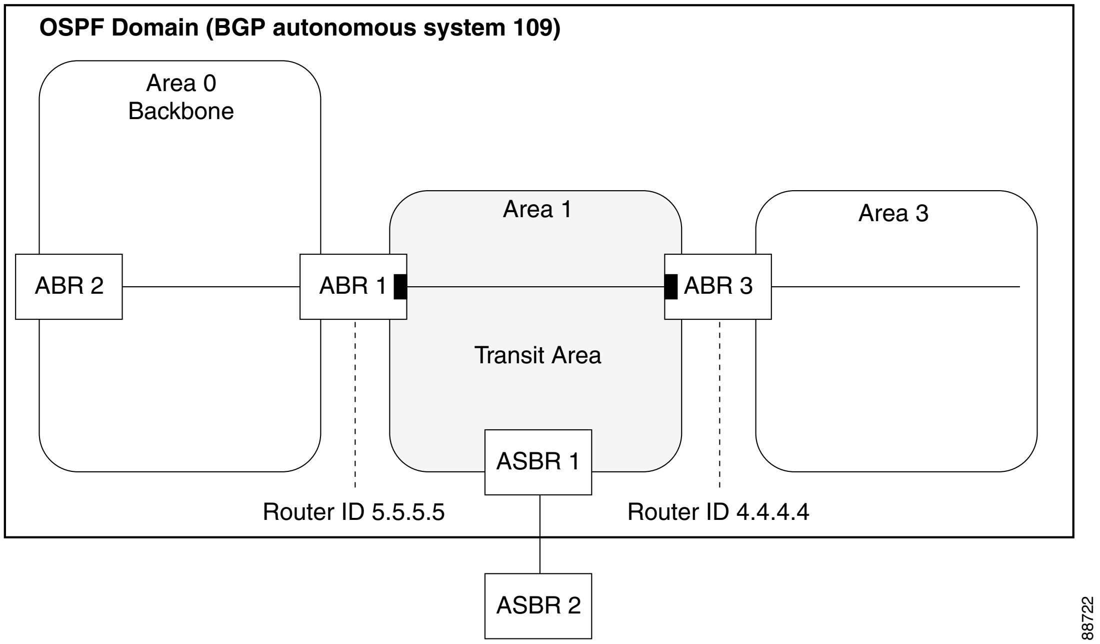
In OSPF, routing information from all areas is first summarized to the backbone area by ABRs. The same ABRs, in turn, propagate such received information to their attached areas. Such hierarchical distribution of routing information requires that all areas be connected to the backbone area (Area 0). Occasions might exist for which an area must be defined, but it cannot be physically connected to Area 0. Examples of such an occasion might be if your company makes a new acquisition that includes an OSPF area, or if Area 0 itself is partitioned.

In the case in which an area cannot be connected to Area 0, you must configure a virtual link between that area and Area 0. The two endpoints of a virtual link are ABRs, and the virtual link must be configured in both routers. The common nonbackbone area to which the two routers belong is called a transit area. A virtual link specifies the transit area and the router ID of the other virtual endpoint (the other ABR).

A virtual link cannot be configured through a stub area or NSSA.

Figure 7 illustrates a virtual link from Area 3 to Area 0.

Figure 7 Virtual Link to Area 0



Route Redistribution for OSPF

Redistribution allows different routing protocols to exchange routing information. This technique can be used to allow connectivity to span multiple routing protocols. It is important to remember that the **redistribute** command controls redistribution *into* an OSPF process and not from OSPF. See the "Configuration Examples for Implementing OSPF on Cisco IOS XR Software" section on page RC-187 for an example of route redistribution for OSPF.

OSPF Shortest Path First Throttling

OSPF SPF throttling makes it possible to configure SPF scheduling in millisecond intervals and to potentially delay SPF calculations during network instability. SPF is scheduled to calculate the Shortest Path Tree (SPT) when there is a change in topology. One SPF run may include multiple topology change events.

The interval at which the SPF calculations occur is chosen dynamically and based on the frequency of topology changes in the network. The chosen interval is within the boundary of the user-specified value ranges. If network topology is unstable, SPF throttling calculates SPF scheduling intervals to be longer until topology becomes stable.

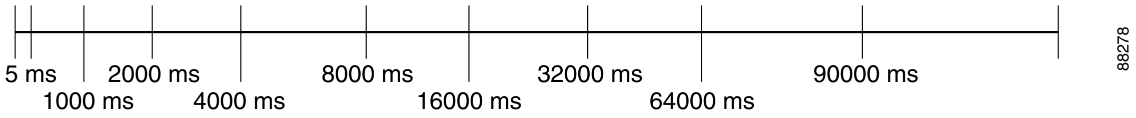
SPF calculations occur at the interval set by the **timers throttle spf** command. The wait interval indicates the amount of time to wait until the next SPF calculation occurs. Each wait interval after that calculation is twice as long as the previous interval until the interval reaches the maximum wait time specified.

The SPF timing can be better explained using an example. In this example, the start interval is set at 5 milliseconds (ms), initial wait interval at 1000 ms, and maximum wait time at 90,000 ms.

```
timers spf 5 1000 90000
```

Figure 8 shows the intervals at which the SPF calculations occur as long as at least one topology change event is received in a given wait interval.

Figure 8 SPF Calculation Intervals Set by the timers spf Command

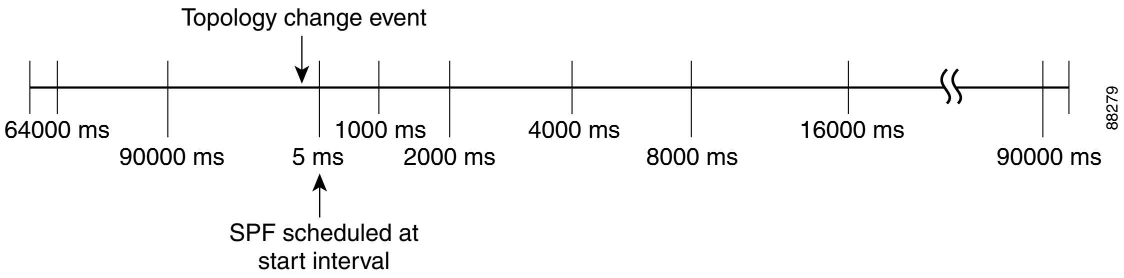


Notice that the wait interval between SPF calculations doubles when at least one topology change event is received during the previous wait interval. After the maximum wait time is reached, the wait interval remains the same until the topology stabilizes and no event is received in that interval.

If the first topology change event is received after the current wait interval, the SPF calculation is delayed by the amount of time specified as the start interval. The subsequent wait intervals continue to follow the dynamic pattern.

If the first topology change event occurs after the maximum wait interval begins, the SPF calculation is again scheduled at the start interval and subsequent wait intervals are reset according to the parameters specified in the **timers throttle spf** command. Notice in Figure 9 that a topology change event was received after the start of the maximum wait time interval and that the SPF intervals have been reset.

Figure 9 Timer Intervals Reset After Topology Change Event



Nonstop Forwarding for OSPF Version 2

Cisco IOS XR NSF for OSPF Version 2 allows for the forwarding of data packets to continue along known routes while the routing protocol information is being restored following a failover. With NSF, peer networking devices do not experience routing flaps. During failover, data traffic is forwarded through intelligent line cards while the standby Route Processor (RP) assumes control from the failed RP. The ability of line cards to remain up through a failover and to be kept current with the Forwarding Information Base (FIB) on the active RP is key to Cisco IOS XR NSF operation.

Routing protocols, such as OSPF, run only on the active RP or DRP and receive routing updates from their neighbor routers. When an OSPF NSF-capable router performs an RP failover, it must perform two tasks to resynchronize its link-state database with its OSPF neighbors. First, it must relearn the available OSPF neighbors on the network without causing a reset of the neighbor relationship. Second, it must reacquire the contents of the link-state database for the network.

As quickly as possible after an RP failover, the NSF-capable router sends an OSPF NSF signal to neighboring NSF-aware devices. This signal is in the form of a link-local LSA generated by the failed-over router. Neighbor networking devices recognize this signal as a cue that the neighbor relationship with this router should not be reset. As the NSF-capable router receives signals from other routers on the network, it can begin to rebuild its neighbor list.

After neighbor relationships are re-established, the NSF-capable router begins to resynchronize its database with all of its NSF-aware neighbors. At this point, the routing information is exchanged between the OSPF neighbors. After this exchange is completed, the NSF-capable device uses the routing information to remove stale routes, update the RIB, and update the FIB with the new forwarding information. OSPF on the router as well as the OSPF neighbors are now fully converged.

Note The standardized IETF version of NSF, known as OSPF graceful restart (RFC 3623) is also supported.

Load Balancing in OSPF Version 2 and OSPFv3

When a router learns multiple routes to a specific network by using multiple routing processes (or routing protocols), it installs the route with the lowest administrative distance in the routing table. Sometimes the router must select a route from among many learned by using the same routing process with the same administrative distance. In this case, the router chooses the path with the lowest cost (or metric) to the destination. Each routing process calculates its cost differently; the costs may need to be manipulated to achieve load balancing.

OSPF performs load balancing automatically. If OSPF finds that it can reach a destination through more than one interface and each path has the same cost, it installs each path in the routing table. The only restriction on the number of paths to the same destination is controlled by the **maximum-paths** (OSPF) command. The default number of maximum paths is 32 for Cisco CRS-1 routers and 16 for Cisco XR 12000 Series Routers. The range is from 1 to 32 for Cisco CRS-1 routers and 1 to 16 for Cisco XR 12000 Series Routers.

Graceful Restart for OSPFv3

In the current release, various restart scenarios in the control plane of an IPv6-enabled router can disrupt data forwarding. The OSPFv3 Graceful Restart feature can preserve the data plane capability in the following circumstances:

- RP failure, resulting in a switchover to the backup processor
- Planned OSPFv3 process restart, such as software upgrade or downgrade
- Unplanned OSPFv3 process restart, such as a process crash

This feature supports non-stop data forwarding on established routes while the OSPFv3 routing protocol is restarting. (Therefore, this feature enhances high availability of IPv6 forwarding.)

Modes of Graceful Restart Operation

The two operational modes that a router can be in for this feature are restart mode and helper mode. Restart mode occurs when the OSPFv3 process is doing a graceful restart. Helper mode refers to the neighbor routers that continue to forward traffic on established OSPFv3 routes while OSPFv3 is restarting on a neighboring router.

Restart Mode

When the OSPFv3 process starts up, it determines whether it must attempt a graceful restart. The determination is based on whether graceful restart was previously enabled. (OSPFv3 does not attempt a graceful restart upon the first-time startup of the router.) When OSPFv3 graceful restart is enabled, it changes the purge timer in the RIB to a non-zero value. See Configuring OSPFv3 Graceful Restart, page RC-181, for descriptions of how to enable and configure the Graceful Restart feature.

During a graceful restart, the router does not populate OSPFv3 routes in the RIB. It tries to bring up full adjacencies with the fully-adjacent neighbors that OSPFv3 had before the restart. Eventually, the OSPFv3 process indicates to the RIB that it has converged either for the purpose of terminating the graceful restart (for any reason) or because it has completed the graceful restart.

The following are general details about restart mode. More detailed information on behavior and certain restrictions and requirements appear in the Graceful Restart Requirements and Restrictions section.

- If the OSPFv3 attempts a restart too soon after the most recent restart, the OSPFv3 process is most likely crashing repeatedly, so the new graceful restart stops running. To control the period between allowable graceful restarts, use the **graceful-restart interval** command. A description of how to set this time period appears in the section Configuring the Minimum Time Required Between Restarts, page RC-183.
- When OSFPv3 starts a graceful restart with the first interface that comes up, a timer starts running to limit the duration (or lifetime) of the graceful restart. You can configure this period with the **graceful-restart lifetime** command. On each interface that comes up, a *grace* LSA (type 11) is flooded to indicate to the neighboring routers that this router is attempting graceful restart. The neighbors enter into helper mode.
- The designated router and backup designated router check of the hello packet received from the restarting neighbor is bypassed because it might not be valid.

Helper Mode

Helper mode is enabled by default. When a (helper) router receives a grace LSA (type 11) from a router that is attempting a graceful restart, the following events occur:

- If helper mode has been disabled through the **graceful-restart helper disable** command, the router drops the LSA packet.
- If helper mode is enabled, the router enters helper mode if all of the following conditions are met.
  - The local router itself is not attempting a graceful restart.
  - The local (helping) router has full adjacency with the sending neighbor.
  - The value of *lsage* (link state age) in the received LSA is less than the requested grace period.
  - The sender of the grace LSA is the same as the originator of the grace LSA.

- Upon entering helper mode, a router performs its helper function for a specific period of time. This time period is the lifetime value from the router that is in restart mode—minus the value of *lsage* in the received grace LSA. If the graceful restart succeeds in time, the helper's timer is stopped before it expires. If the helper's timer does expire, the adjacency to the restarting router is brought down, and normal OSPFv3 functionality resumes.
- The dead timer is not honored by the router that is in helper mode.
- A router in helper mode ceases to perform the helper function in any of the following cases:
  - The helper router is able to bring up a FULL adjacency with the restarting router.
  - The local timer for the helper function expires.

Graceful Restart Requirements and Restrictions

The requirements for supporting the Graceful Restart feature include:

- Cooperation of a router's neighbors during a graceful restart. In relation to the router on which OSPFv3 is restarting, each router is called a *helper.*
- All neighbors of the router that does a graceful restart must be capable of doing a graceful restart.
- A graceful restart does not occur upon the first-time startup of a router.
- OSPFv3 neighbor information and database information are not check-pointed.
- An OSPFv3 process rebuilds adjacencies after it restarts.
- To ensure consistent databases after a restart, the OSPFv3 configuration must be identical to the configuration before the restart. (This requirement applies to self-originated information in the local database.) A graceful restart can fail if configurations change during the operation. In this case, data forwarding would be affected. OSPFv3 resumes operation by regenerating all its LSAs and resynchronizing its database with all its neighbors.
- Although IPv6 FIB tables remain unchanged during a graceful restart, these tables eventually mark the routes as stale through the use of a holddown timer. Enough time is allowed for the protocols to rebuild state information and converge.
- The router on which OSPFv3 is restarting must send OSPFv3 hellos within the dead interval of the process restart. Protocols must be able to retain adjacencies with neighbors before the adjacency dead timer expires. The default for the dead timer is 40 seconds. If hellos do not arrive on the adjacency before the dead timer expires, the router takes down the adjacency. The OSPFv3 Graceful Restart feature does not function properly if the dead timer is configured to be less than the time required to send hellos after the OSPFv3 process restarts.
- Simultaneous graceful restart sessions on multiple routers are not supported on a single network segment. If a router determines that multiple routers are in restart mode, it terminates any local graceful restart operation.
- This feature utilizes the available support for changing the purge time of existing OSPFv3 routes in the routing information base (RIB). When graceful restart is enabled, the purge timer is set to 90 seconds by default. If graceful restart is disabled, the purge timer setting is 0.
- This feature has an associated *grace* LSA. This link-scope LSA is type 11.
- According to the RFC, the OSPFv3 process should flush all old, self-originated LSAs during a restart. With the Graceful Restart feature, however, the router delays this flushing of unknown self-originated LSAs during a graceful restart. OSPFv3 can learn new information and build new LSAs to replace the old LSAs. When the delay is over, all old LSAs are flushed.

- If graceful restart is enabled, the adjacency creation time of all the neighbors is saved in the system database (SysDB). The purpose for saving the creation time is so that OSPFv3 can use the original adjacency creation time to display the uptime for that neighbor after the restart.

Multicast-Intact Feature

The multicast-intact feature provides the ability to run multicast routing (PIM) when IGP shortcuts are configured and active on the router. Both OSPFv2 and IS-IS support the multicast-intact feature.

You can enable multicast-intact in the IGP when multicast routing protocols (PIM) are configured and IGP shortcuts are configured on the router. IGP shortcuts are MPLS tunnels that are exposed to IGP. The IGPs routes IP traffic over these tunnels to destinations that are downstream from the egress router of the tunnel (from an SPF perspective). PIM cannot use IGP shortcuts for propagating PIM joins because reverse path forwarding (RPF) cannot work across a unidirectional tunnel.

When you enable multicast-intact on an IGP, the IGP publishes a parallel or alternate set of equal-cost next-hops for use by PIM. These next-hops are called *mcast-intact* next-hops. The mcast-intact next-hops have the following attributes:

- They are guaranteed not to contain any IGP shortcuts.
- They are not used for unicast routing but are used only by PIM to look up an IPv4 next-hop to a PIM source.
- They are not published to the FIB.
- When multicast-intact is enabled on an IGP, all IPv4 destinations that were learned through link-state advertisements are published with a set equal-cost mcast-intact next-hops to the RIB. This attribute applies even when the native next-hops have no IGP shortcuts.
- In OSPF, the max-paths (number of equal-cost next-hops) limit is applied separately to the native and mcast-intact next-hops. The number of equal cost mcast-intact next-hops is the same as that configured for the native next-hops. (In IS-IS, the behavior is slightly different.)

How to Implement OSPF on Cisco IOS XR Software

This section contains the following procedures:

- Enabling OSPF, page RC-145 (required)
- Configuring Stub and Not-so-Stubby Area Types, page RC-147 (optional)
- Configuring Neighbors for Nonbroadcast Networks, page RC-150 (optional)
- Configuring Authentication at Different Hierarchical Levels for OSPF Version 2, page RC-155 (optional)
- Controlling the Frequency that the Same LSA Is Originated or Accepted for OSPF, page RC-158 (optional)
- Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF, page RC-160 (optional)
- Summarizing Subnetwork LSAs on an OSPF ABR, page RC-164 (optional)
- Redistributing Routes from One IGP into OSPF, page RC-166 (optional)
- Configuring OSPF Shortest Path First Throttling, page RC-170 (optional)
- Configuring Nonstop Forwarding for OSPF Version 2, page RC-173 (optional)
- Configuring OSPF Version 2 for MPLS Traffic Engineering, page RC-175 (optional)

- Verifying OSPF Configuration and Operation, page RC-180 (optional)
- Configuring OSPFv3 Graceful Restart, page RC-181 (optional)
- Enabling Multicast-Intact for OSPFv2, page RC-186 (optional)

Enabling OSPF

This task explains how to perform the minimum OSPF configuration on your router that is to enable an OSPF process with a router ID, configure a backbone or nonbackbone area, and then assign one or more interfaces on which OSPF runs.

Prerequisites

Although you can configure OSPF before you configure an IP address, no OSPF routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **area** *area-id*
5. **interface** *type instance*
6. Repeat Step 5 for each interface that use OSPF.
7. **log adjacency changes** [**detail**] [**enable** | **disable**]
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router ospf** *process-name*
	or
	router ospfv3 *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** {*ipv4-address* | *interface-type interface-instance*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IP address as the router ID.

Step 4	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures an area for the OSPF process. • Backbone areas have an area ID of 0. • Nonbackbone areas have a nonzero area ID. • The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 5	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/1/0/3`	Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4.
Step 6	Repeat Step 5 for each interface that uses OSPF.	—

	Command or Action	Purpose
Step 7	**log adjacency changes** [**detail**] [**enable** | **disable**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# log adjacency changes detail`	(Optional) Requests notification of neighbor changes. • By default, this feature is enabled. • The messages generated by neighbor changes are considered notifications, which are categorized as severity Level 5 in the **logging console** command. The **logging console** command controls which severity level of messages are sent to the console. By default, all severity level messages are sent.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Stub and Not-so-Stubby Area Types

This task explains how to configure the stub area and the NSSA for OSPF.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **area** *area-id*
5. **stub** [**no-summary**]
   or
   nssa [**no-redistribution**] [**default-information-originate**] [**no-summary**]

6. **stub**
   or
   nssa
7. **default-cost** *cost*
8. **end**
   or
   commit
9. Repeat this task on all other routers in the stub area or NSSA.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	or
	`router ospfv3` *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** `{`*`ipv4-address`* `|` *`interface-type interface-instance`*`}`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IP address as the router ID.

Step 4	**`area`** *`area-id`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# area 1`
		Enters area configuration mode and configures a nonbackbone area for the OSPF process.
		• The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

	Command or Action	Purpose

Step 5	**stub** [**no-summary**]
	or
	nssa [**no-redistribution**] [**default-information-originate**] [**no-summary**]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# stub no summary`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# nssa no-redistribution`
		Defines the nonbackbone area as a stub area.
		• See the "Configuring Stub and Not-so-Stubby Area Types" section on page RC-147.
		• Specify the **no-summary** keyword to further reduce the number of LSAs sent into a stub area. This keyword prevents the ABR from sending summary link-state advertisements (Type 3) in the stub area.
		or
		Defines an area as an NSSA.
		• See the "Configuring Stub and Not-so-Stubby Area Types" section on page RC-147.

Step 6	**stub**
	or
	nssa
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# stub`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# nssa`
		(Optional) Turns off the options configured for stub and NSSA areas.
		• If you configured the stub and NSSA areas using the optional keywords (**no-summary**, **no-redistribution**, **default-information-originate**, and **no-summary**) in Step 5, you must now reissue the **stub** and **nssa** commands without the keywords—rather than using the **no** form of the command.
		• For example, the **no nssa default-information-originate** form of the command changes the NSSA area into a normal area that inadvertently brings down the existing adjacencies in that area.

Step 7	**default-cost** *cost* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# default-cost 15`	(Optional) Specifies a cost for the default summary route sent into a stub area or an NSSA. • Use this command only on ABRs attached to the NSSA. Do not use it on any other routers in the area. • The default cost is 1.

	Command or Action	Purpose

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	Repeat this task on all other routers in the stub area or NSSA.	—

Configuring Neighbors for Nonbroadcast Networks

This task explains how to configure neighbors for a nonbroadcast network. This task is optional.

Prerequisites

Configuring NBMA networks as either broadcast or nonbroadcast assumes that there are virtual circuits from every router to every other router or a fully meshed network.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **area** *area-id*
5. **network** {**broadcast** | **non-broadcast** | {**point-to-multipoint** [**non-broadcast**] | **point-to-point**}}
6. **dead-interval** *seconds*
7. **hello-interval** *seconds*
8. **interface** *type number*

9. **neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*] [**cost** *number*]
or
neighbor *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*] [**cost** *number*] [**database-filter** [**all**]]
10. Repeat Step 9 for all neighbors on the interface.
11. **exit**
12. **interface** *type instance*
13. **neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]
or
neighbor *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]
14. Repeat Step 13 for all neighbors on the interface.
15. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	or
	`router ospfv3` *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** `{`*`ipv4-address`* `|` *`interface-type interface-instance`*`}`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IP address as the router ID.

Step 4	**`area`** *`area-id`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# area 0`
		Enters area configuration mode and configures an area for the OSPF process.
		• This example configures a backbone area.
		• The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

	Command or Action	Purpose

Step 5	**network** {**broadcast** | **non-broadcast** | {**point-to-multipoint** [**non-broadcast**] | **point-to-point**}}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# network non-broadcast`
		Configures the OSPF network type to a type other than the default for a given medium.
		• The example sets the network type to NBMA.

Step 6	**dead-interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# dead-interval 40`	(Optional) Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.
Step 7	**hello-interval** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# hello-interval 10`	(Optional) Specifies the interval between hello packets that OSPF sends on the interface.
Step 8	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/2/0/0`	Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4. • In this example, the interface inherits the nonbroadcast network type and the hello and dead intervals from the areas because the values are not set at the interface level.

	Command or Action	Purpose

Step 9	**neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*]
	or
	neighbor *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor 10.20.20.1 priority 3 poll-interval 15`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# neighbor fe80::3203:a0ff:fe9d:f3fe`
		Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.
		or
		Configures the link-local IPv6 address of OSPFv3 neighbors.
		• The *ipv6-link-local-address* must be in the form that is specified in RFC 2373. The address is specified in hexadecimal using 16-bit values between colons.
		• The **priority** keyword notifies the router that this neighbor is eligible to become a DR or BDR. The priority value should match the actual priority setting on the neighbor router. The neighbor priority default value is 0. This keyword does not apply to point-to-multipoint interfaces.
		• The **poll-interval** keyword does not apply to point-to-multipoint interfaces. RFC 1247 recommends that this value be much larger than the hello interval. The default is 120 seconds (2 minutes).
		• Neighbors with no specific cost configured assumes the cost of the interface, based on the **cost** command. On point-to-multipoint interfaces, **cost** *number* is the only keyword and argument combination that works. The **cost** keyword does not apply to NBMA networks.
		• The **database-filter** keyword filters outgoing LSAs to an OSPF neighbor. If you specify the **all** keyword, incoming and outgoing LSAs are filtered. Use with extreme caution because filtering might cause the routing topology to be seen as entirely different between two neighbors, resulting in black-holing of data traffic or routing loops.

Step 10	Repeat Step 9 for all neighbors on the interface.	—
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# exit`	Enters area configuration mode.
Step 12	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/3/0/1`	Enters interface configuration mode and associates one or more interfaces for the area configured in Step 4. • In this example, the interface inherits the nonbroadcast network type and the hello and dead intervals from the areas because the values are not set at the interface level.

	Command or Action	Purpose

Step 13	**neighbor** *ip-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]
	or
	neighbor *ipv6-link-local-address* [**priority** *number*] [**poll-interval** *seconds*][**cost** *number*] [**database-filter** [**all**]]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# neighbor 10.34.16.6`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# neighbor fe80::3203:a0ff:fe9d:f3f`
		Configures the IPv4 address of OSPF neighbors interconnecting to nonbroadcast networks.
		or
		Configures the link-local IPv6 address of OSPFv3 neighbors.
		• The *ipv6-link-local-address* argument must be in the form documented in RFC 2373 in which the address is specified in hexadecimal using 16-bit values between colons.
		• The **priority** keyword notifies the router that this neighbor is eligible to become a DR or BDR. The priority value should match the actual priority setting on the neighbor router. The neighbor priority default value is zero. This keyword does not apply to point-to-multipoint interfaces.
		• The **poll-interval** keyword does not apply to point-to-multipoint interfaces. RFC 1247 recommends that this value be much larger than the hello interval. The default is 120 seconds (2 minutes).
		• Neighbors with no specific cost configured assumes the cost of the interface, based on the **cost** command. On point-to-multipoint interfaces, **cost** *number* is the only keyword and argument combination that works. The **cost** keyword does not apply to NBMA networks.
		• The **database-filter** keyword filters outgoing LSAs to an OSPF neighbor. If you specify the **all** keyword, incoming and outgoing LSAs are filtered. Use with extreme caution since filtering may cause the routing topology to be seen as entirely different between two neighbors, resulting in ‘black-holing’ or routing loops.

	Command or Action	Purpose
Step 14	Repeat Step 13 for all neighbors on the interface.	—

Step 15	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Authentication at Different Hierarchical Levels for OSPF Version 2

This task explains how to configure MD5 (secure) authentication on the OSPF router process, configure one area with plain text authentication, and then apply one interface with clear text (null) authentication.

Note Authentication configured at the interface level overrides authentication configured at the area level and the router process level. If an interface does not have authentication specifically configured, the interface inherits the authentication parameter value from a higher hierarchical level. See the "OSPF Hierarchical CLI and CLI Inheritance" section on page RC-131 for more information about hierarchy and inheritance.

Prerequisites

If you choose to configure authentication, you must first decide whether to configure plain text or MD5 authentication, and whether the authentication applies to all interfaces in a process, an entire area, or specific interfaces. See the "Route Authentication Methods for OSPF Version 2" section on page RC-135 for information about each type of authentication and when you should use a specific method for your network.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **authentication** [**message-digest** | **null**]

5. **message-digest-key** *key-id* **md5** {*key* | **clear** *key* | **encrypted** *key*}
6. **area** *area-id*
7. **interface** *type instance*
8. Repeat Step 7 for each interface that must communicate, using the same authentication.
9. **exit**
10. **area** *area-id*
11. **authentication** [**message-digest** | **null**]
12. **interface** *type instance*
13. Repeat Step 7 for each interface that must communicate, using the same authentication.
14. **interface** *type instance*
15. **authentication** [**message-digest** | **null**]
16. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** {*`ipv4-address`* | *`interface-type interface-instance`*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.

Step 4	**`authentication`** [**`message-digest`** | **`null`**] **Example:** `RP/0/RP0/CPU0:router(config-ospf)# authentication message-digest`	Enables MD5 authentication for the OSPF process. • This authentication type applies to the entire router process unless overridden by a lower hierarchical level such as the area or interface.

Step 5	**`message-digest-key`** *`key-id`* **`md5`** {*`key`* | **`clear`** *`key`* | **`encrypted`** *`key`*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# message-digest-key 4 md5 yourkey`
		Specifies the MD5 authentication key for the OSPF process.
		• The neighbor routers must have the same key identifier.

	Command or Action	Purpose
Step 6	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures a backbone area for the OSPF process.
Step 7	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/1/0/3`	Enters interface configuration mode and associates one or more interfaces to the backbone area. • All interfaces inherit the authentication parameter values specified for the OSPF process (Step 4, Step 5, and Step 6).
Step 8	Repeat Step 7 for each interface that must communicate, using the same authentication.	—
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# exit`	Enters area OSPF configuration mode.
Step 10	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures a nonbackbone area 1 for the OSPF process. • The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 11	**authentication** [**message-digest** | **null**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# authentication`	Enables Type 1 (plain text) authentication that provides no security. • The example specifies plain text authentication (by not specifying a keyword). Use the **authentication-key** interface command to specify the plain text password.
Step 12	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/1/0/0`	Enters interface configuration mode and associates one or more interfaces to the nonbackbone area 1 specified in Step 7. • All interfaces configured inherit the authentication parameter values configured for area 1.
Step 13	Repeat Step 12 for each interface that must communicate, using the same authentication.	—
Step 14	**interface** *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/3/0/0`	Enters interface configuration mode and associates one or more interfaces to a different authentication type.

	Command or Action	Purpose
Step 15	**authentication** [**message-digest** | **null**] **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# authentication null`	Specifies no authentication on POS interface 0/3/0/0, overriding the plain text authentication specified for area 1. • By default, all of the interfaces configured in the same area inherit the same authentication values of the area.

Step 16	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Controlling the Frequency that the Same LSA Is Originated or Accepted for OSPF

This task explains how to tune the convergence time of OSPF routes in the routing table when many LSAs need to be flooded in a very short time interval.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. Do Step 5, Step 6 or both to control the frequency that the same LSA is originated or accepted.
5. **timers lsa gen-interval** *seconds*
6. **timers lsa min-arrival** *seconds*
7. **timers lsa group-pacing** *seconds*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	**router ospf** *process-name* or **router ospfv3** *process-name* **Example:** RP/0/RP0/CPU0:router(config)# router ospf 1 or RP/0/RP0/CPU0:router(config)# router ospfv3 1	Enables OSPF routing for the specified routing process, and places the router in router configuration mode. or Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.
Step 3	**router-id** {*ipv4-address* | *interface-type interface-instance*} **Example:** RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3	Configures a router ID for the OSPF process. **Note** We recommend using a stable IP address as the router ID.
Step 4	Perform Step 5 or Step 6 or both to control the frequency that the same LSA is originated or accepted.	—
Step 5	**timers lsa gen-interval** *seconds* **Example:** RP/0/RP0/CPU0:router(config-ospf)# timers lsa gen-interval 10	Changes the minimum interval between the same OSPF LSAs that the router originates. • The default is 5 seconds for both OSPF and OSPFv3.
Step 6	**timers lsa min-arrival** *seconds* **Example:** RP/0/RP0/CPU0:router(config-ospf)# timers lsa min-arrival 2	Limits the frequency that new processes of any particular OSPF Version 2 LSA can be accepted during flooding. • The default is 1 second.

	Command or Action	Purpose

Step 7	**`timers lsa group-pacing`** *`seconds`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# timers lsa group-pacing 1000`
		Changes the interval at which OSPF link-state LSAs are collected into a group for flooding. The default is 240 seconds.

Step 8	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Creating a Virtual Link with MD5 Authentication to Area 0 for OSPF

This task explains how to create a virtual link to your backbone (area 0) and apply MD5 authentication. You must perform the steps described on both ABRs, one at each end of the virtual link. To understand virtual links, see the "Virtual Link and Transit Area for OSPF" section on page RC-138.

Note After you explicitly configure area parameter values, they are inherited by all interfaces bound to that area—unless you override the values and configure them explicitly for the interface. An example is provided in the "Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example" section on page RC-192.

Prerequisites

Meet the following prerequisites before you create a virtual link with MD5 authentication to area 0:

- Have the router ID of the neighbor router at the opposite end of the link to configure the local router. You can use the **show ospf** or **show ospfv3** command on the remote end to get its router ID.

- For a virtual link to be successful, you need a stable router ID at each end of the virtual link. You do not want them to be subject to change, which could happen if they are assigned by default (See the "OSPF Process and Router ID" section on page RC-134 for an explanation of how the router ID is determined.) Therefore, we recommend that you perform one of the following tasks before configuring a virtual link:
  - Use the **router-id** command to set the router ID. This strategy is preferable.
  - Configure a loopback interface so that the router has a stable router ID.
- Before configuring your virtual link for OSPF Version 2, you must decide whether to configure plain text authentication, MD5 authentication, or no authentication (which is the default). Your decision determines whether you need to perform additional tasks related to authentication.

Note If you decide to configure plain text authentication or no authentication, see the **authentication** command provided in the *OSPF Commands on Cisco IOS XR Software* module in the *Cisco IOS XR Routing Command Reference*.

SUMMARY STEPS

1. **show ospf** [*process-name*]
   or
   show ospfv3 [*process-name*]
2. **configure**
3. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
4. **router-id** {*ipv4-address* | *interface-type interface-instance*}
5. **area** *area-id*
6. **virtual link** *router-id*
7. **authentication message-digest**
8. **message-digest-key** *key-id* **md5** {*key* | **clear** *key* | **encrypted** *key*}
9. Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.
10. **end**
    or
    commit
11. **show ospf** [*process-name*] [*area-id*] **virtual-links**
    or
    show ospfv3 [*process-name*] **virtual-links**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**show ospf** [*process-name*]
	or
	show ospfv3 [*process-name*]
	Example:
	`RP/0/RP0/CPU0:router# show ospf`
	or
	`RP/0/RP0/CPU0:router# show ospfv3`
		(Optional) Displays general information about OSPF routing processes.
		- The output displays the router ID of the local router. You need this router ID to configure the other end of the link.

Step 2	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 3	**router ospf** *process-name*
	or
	router ospfv3 *process-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 4	**router-id** {*ipv4-address* | *interface-type interface-instance*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

Step 5	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode and configures a nonbackbone area for the OSPF process. - The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.
Step 6	**virtual-link** *router-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# virtual link 10.3.4.5`	Defines an OSPF virtual link. - See the "Virtual Link and Transit Area for OSPF" section on page RC-138.
Step 7	**authentication message-digest** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-vl)# authentication message-digest`	Selects MD5 authentication for this virtual link.

	Command or Action	Purpose

Step 8	**message-digest-key** *key-id* **md5** {*key* | **clear** *key* | **encrypted** *key*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-vl)# message-digest-key 4 md5 yourkey`
		Defines an OSPF virtual link.
		• See the "Virtual Link and Transit Area for OSPF" section on page RC-138 to understand a virtual link.
		• The *key-id* argument is a number in the range from 1 to 255. The *key* argument is an alphanumeric string of up to 16 characters. The routers at both ends of the virtual link must have the same key identifier and key to be able to route OSPF traffic.
		• The **authentication-key** *key* command is not supported for OSPFv3.
		• Once the key is encrypted it must remain encrypted.

Step 9	Repeat all of the steps in this task on the ABR that is at the other end of the virtual link. Specify the same key ID and key that you specified for the virtual link on this router.	—

Step 10	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-vl)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-vl)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show ospf** [*process-name*] [*area-id*] **virtual-links**
	or
	show ospfv3 [*process-name*] **virtual-links**
	Example:
	`RP/0/RP0/CPU0:router# show ospf 1 2 virtual-links`
	or
	`RP/0/RP0/CPU0:router# show ospfv3 1 virtual-links`
		(Optional) Displays the parameters and the current state of OSPF virtual links.

Examples

In the following example, the **show ospfv3 virtual links** EXEC command verifies that the OSPF_VL0 virtual link to the OSPFv3 neighbor is up, the ID of the virtual link interface is 2, and the IPv6 address of the virtual link endpoint is 2003:3000::1.

```
RP/0/RP0/CPU0:router# show ospfv3 virtual-links

Virtual Links for OSPFv3 1

Virtual Link OSPF_VL0 to router 10.0.0.3 is up
  Interface ID 2, IPv6 address 2003:3000::1
  Run as demand circuit
  DoNotAge LSA allowed.
  Transit area 0.1.20.255, via interface POS 0/1/0/1, Cost of using 2
  Transmit Delay is 5 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:02
    Adjacency State FULL (Hello suppressed)
    Index 0/2/3, retransmission queue length 0, number of retransmission 1
    First 0(0)/0(0)/0(0) Next 0(0)/0(0)/0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec

Check for lines:
Virtual Link OSPF_VL0 to router 10.0.0.3 is up
    Adjacency State FULL (Hello suppressed)

State is up and Adjacency State is FULL
```

Summarizing Subnetwork LSAs on an OSPF ABR

If you configured two or more subnetworks when you assigned your IP addresses to your interfaces, you might want the software to summarize (aggregate) into a single LSA all of the subnetworks that the local area advertises to another area. Such summarization would reduce the number of LSAs and thereby conserve network resources. This summarization is known as interarea route summarization. It applies to routes from within the autonomous system. It does not apply to external routes injected into OSPF by way of redistribution.

This task configures OSPF to summarize subnetworks into one LSA, by specifying that all subnetworks that fall into a range are advertised together. This task is performed on an ABR only.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **area** *area-id*
5. **range** *ip-address mask* [**advertise** | **not-advertise**]
   or
   range *ipv6-prefix*/*prefix-length* [**advertise** | **not-advertise**]
6. **interface** *type instance*

7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	or
	`router ospfv3` *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** `{`*`ipv4-address`* `|` *`interface-type interface-instance`*`}`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

Step 4	**`area`** *`area-id`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# area 0`
		Enters area configuration mode and configures a nonbackbone area for the OSPF process.
		• The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 5	**`range`** *`ip-address mask`* `[`**`advertise`** `|` **`not-advertise`**`]`
	or
	`range` *`ipv6-prefix`*`/`*`prefix-length`* `[`**`advertise`** `|` **`not-advertise`**`]`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# range 192.168.0.0 255.255.0.0 advertise`
	or
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# range 4004:f000::/32 advertise`
		Consolidates and summarizes OSPF routes at an area boundary.
		• The **advertise** keyword causes the software to advertise the address range of subnetworks in a Type 3 summary LSA.
		• The **not-advertise** keyword causes the software to suppress the Type 3 summary LSA, and the subnetworks in the range remain hidden from other areas.
		• In the first example, all subnetworks for network 192.168.0.0 are summarized and advertised by the ABR into areas outside the backbone.
		• In the second example, two or more IPv4 interfaces are covered by a 192.*x.x* network.

	Command or Action	Purpose

Step 6	`interface` *`type instance`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/2/0/3`
		Enters interface configuration mode and associates one or more interfaces to the area.

Step 7	`end`
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Redistributing Routes from One IGP into OSPF

This task redistributes routes from an IGP (could be a different OSPF process) into OSPF.

Prerequisites

For information about configuring routing policy, see the *Implementing Routing Policy on Cisco IOS XR Software* module.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **redistribute** *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match** {**internal** | **external** [**1** | **2**} | **nssa-external** [**1** | **2**}] [**tag** *tag-value*] [**route-map** *map-tag* | **policy** *policy-tag*]

5. **summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
   or
   summary-prefix *ipv6-prefix/prefix-length* [**not-advertise**] [**tag** *tag*]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	or
	`router ospfv3` *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** `{`*`ipv4-address`* `|` *`interface-type interface-instance`*`}`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

	Command or Action	Purpose

Step 4	**redistribute** *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match** {**internal** | **external** [**1** | **2**} | **nssa-external** [**1** | **2**}] [**tag** *tag-value*] [**route-map** *map-tag* | **policy** *policy-tag*]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# redistribute bgp 1 level-1`
	or
	`RP/0/RP0/CPU0:router(config-router)# redistribute bgp 1 level-1-2 metric-type 1`
		Redistributes OSPF routes from one routing domain to another routing domain.
		or
		Redistributes OSPFv3 routes from one routing domain to another routing domain.
		• This command causes the router to become an ASBR by definition.
		• OSPF tags all routes learned through redistribution as external.
		• The protocol and its process ID, if it has one, indicate the protocol being redistributed into OSPF.
		• The metric is the cost you assign to the external route. The default is 20 for all protocols except BGP, whose default metric is 1.
		• The OSPF example redistributes BGP autonomous system 1, Level 1 routes into OSPF as Type 2 external routes.
		• The OSPFv3 example redistributes BGP autonomous system 1, Level 1 and 2 routes into OSPF. The external link type associated with the default route advertised into the OSPFv3 routing domain is the Type 1 external route.
		Note RPL is not supported for OSPFv3.

	Command or Action	Purpose

Step 5	**summary-prefix** *address mask* [**not-advertise**] [**tag** *tag*]
	or
	summary-prefix *ipv6-prefix***/***prefix-length* [**not-advertise**] [**tag** *tag*]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# summary-prefix 10.1.0.0 255.255.0.0`
	or
	`RP/0/RP0/CPU0:router(config-router)# summary-prefix 2010:11:22::/32`
		(Optional) Creates aggregate addresses for OSPF.
		or
		(Optional) Creates aggregate addresses for OSPFv3.
		• This command provides external route summarization of the non-OSPF routes.
		• External ranges that are being summarized should be contiguous. Summarization of overlapping ranges from two different routers could cause packets to be sent to the wrong destination.
		• This command is optional. If you do not specify it, each route is included in the link-state database and advertised in LSAs.
		• In the OSPFv2 example, the summary address 10.1.0.0 includes address 10.1.1.0, 10.1.2.0, 10.1.3.0, and so on. Only the address 10.1.0.0 is advertised in an external LSA.
		• In the OSPFv3 example, the summary address 2010:11:22::/32 has addresses such as 2010:11:22:0:1000::1, 2010:11:22:0:2000:679:1, and so on. Only the address 2010:11:22::/32 is advertised in the external LSA.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)?`
		`[cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring OSPF Shortest Path First Throttling

This task explains how to configure SPF scheduling in millisecond intervals and potentially delay SPF calculations during times of network instability. This task is optional.

Prerequisites

See the "OSPF Shortest Path First Throttling" section on page RC-139 for information about OSPF SPF throttling.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
   or
   router ospfv3 *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **timers throttle spf** *spf-start spf-hold spf-max-wait*
5. **area** *area-id*
6. **interface** *type instance*
7. **end**
   or
   commit
8. **show ospf** [*process-name*]
   or
   show ospfv3 [*process-name*]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	or
	`router ospfv3` *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
	or
	`RP/0/RP0/CPU0:router(config)# router ospfv3 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		or
		Enables OSPFv3 routing for the specified routing process, and places the router in router ospfv3 configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

	Command or Action	Purpose

Step 3	**`router-id`** {*`ipv4-address`* | *`interface-type interface-instance`*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

Step 4	**`timers throttle spf`** *`spf-start spf-hold spf-max-wait`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# timers throttle spf 10 4800 90000`
		Sets SPF throttling timers.

Step 5	**`area`** *`area-id`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)#  area 0`
		Enters area configuration mode and configures a backbone area.
		• The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area. We recommend using the IPv4 address notation.

Step 6	**`interface`** *`type instance`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# interface POS 0/1/0/3`
		Enters interface configuration mode and associates one or more interfaces to the area.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show ospf** [*process-name*]
	or
	show ospfv3 [*process-name*]
	Example:
	`RP/0/RP0/CPU0:router# show ospf 1`
	or
	`RP/0/RP0/CPU0:router# show ospfv3 2`
		(Optional) Displays SPF throttling timers.

Examples

In the following example, the **show ospf** EXEC command is used to verify that the initial SPF schedule delay time, minimum hold time, and maximum wait time are configured correctly. Additional details are displayed about the OSPF process, such as the router type and redistribution of routes.

```
RP/0/RP0/CPU0:router# show ospf 1

Routing Process "ospf 1" with ID 192.168.4.3
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 It is an autonomous system boundary router
 Redistributing External Routes from,
    ospf 2
 Initial SPF schedule delay 5 msecs
 Minimum hold time between two consecutive SPFs 100 msecs
 Maximum wait time between two consecutive SPFs 1000 msecs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 00000000
 Number of opaque AS LSA 0. Checksum Sum 00000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 External flood list length 0
```

```
Non-Stop Forwarding enabled
```

Note For a description of each output display field, see the **show ospf** command in the *OSPF Commands on Cisco IOS XR Software* module in the *Cisco IOS XR Routing Command Reference* document.

Configuring Nonstop Forwarding for OSPF Version 2

This task explains how to configure OSPF NSF on your NSF-capable router. This task is optional.

Prerequisites

OSPF NSF requires that all neighbor networking devices be NSF aware, which happens automatically after you install the Cisco IOS XR image on the router. If an NSF-capable router discovers that it has non-NSF-aware neighbors on a particular network segment, it disables NSF capabilities for that segment. Other network segments composed entirely of NSF-capable or NSF-aware routers continue to provide NSF capabilities.

See the "Nonstop Forwarding for OSPF Version 2" section on page RC-140 for conceptual information.

Restrictions

The following are restrictions when configuring nonstop forwarding:

- OSPF Cisco NSF for virtual links is not supported.
- Neighbors must be NSF aware.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **nsf**
   or
   nsf enforce global
5. **nsf interval** *seconds*
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# router ospf 1`
		Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
		Note The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**`router-id`** `{`*`ipv4-address`* `|` *`interface-type interface-instance`*`}`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

Step 4	**`nsf`**
	or
	`nsf enforce global`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# nsf`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# nsf enforce global`
		Enables OSPF NSF operations.
		• Use the **nsf** command without the optional **enforce** and **global** keywords to abort the NSF restart mechanism on the interfaces of detected non-NSF neighbors and allow NSF neighbors to function properly.
		• Use the **nsf** command with the optional **enforce** and **global** keywords if the router is expected to perform NSF during restart. However, if non-NSF neighbors are detected, NSF restart is canceled for the entire OSPF process.

	Command or Action	Purpose

Step 5	**`nsf interval`** *`seconds`*
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# nsf interval 120`
		Sets the minimum time between NSF restart attempts.
		Note When you use this command, the OSPF process must be up for at least 90 seconds before OSPF attempts to perform an NSF restart.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring OSPF Version 2 for MPLS Traffic Engineering

This task explains how to configure OSPF for MPLS TE. This task is optional.

For a description of the MPLS TE tasks and commands that allow you to configure the router to support tunnels, configure an MPLS tunnel that OSPF can use, and troubleshoot MPLS TE, see the *Implementing MPLS Traffic Engineering Configuration Guide.*

Prerequisites

Your network must support the following Cisco IOS XR features before you enable MPLS TE for OSPF on your router:

- MPLS
- IP Cisco Express Forwarding (CEF)

Note You must enter the commands in the following task on every OSPF router in the traffic-engineered portion of your network.

Restrictions

MPLS traffic engineering currently supports only a single OSPF area.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** {*ipv4-address* | *interface-type interface-instance*}
4. **mpls traffic-eng area** *area-id*
5. **mpls traffic-eng router-id** {*ip-address* | *interface-type interface-instance*}
6. **area** *area-id*
7. **interface** *type instance*
8. **end**
   or
   commit
9. **show ospf** [*process-name*] [*area-id*] **mpls traffic-eng** {**link** | **fragment**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing process, and places the router in router configuration mode. **Note** The *process-name* argument is any alphanumeric string no longer than 40 characters.

Step 3	**router-id** {*ipv4-address* | *interface-type interface-instance*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# router-id 192.168.4.3`
		Configures a router ID for the OSPF process.
		Note We recommend using a stable IPv4 address as the router ID.

Step 4	**mpls traffic-eng area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng area 0`	Configures the OSPF area for MPLS TE.

	Command or Action	Purpose

Step 5	`mpls traffic-eng router-id` {*ip-address* | *interface-type interface-instance*}
	Example:
	`RP/0/RP0/CPU0:router(config-ospf)# mpls traffic-eng router-id loopback 0`
		(Optional) Specifies that the traffic engineering router identifier for the node is the IP address associated with a given interface.
		• This IP address is flooded to all nodes in TE LSAs.
		• For all traffic engineering tunnels originating at other nodes and ending at this node, you must set the tunnel destination to the traffic engineering router identifier of the destination node because that is the address that the traffic engineering topology database at the tunnel head uses for its path calculation.
		• We recommend that loopback interfaces be used for MPLS TE router ID because they are more stable than physical interfaces.

Step 6	`area` *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode and configures an area for the OSPF process. • The *area-id* argument can be entered in dotted-decimal or IPv4 address notation, such as area 1000 or area 0.0.3.232. However, you must choose one form or the other for an area.
Step 7	`interface` *type instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface interface loopback0`	Enters interface configuration mode and associates one or more interfaces to the area.

	Command or Action	Purpose

Step 8	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# end`
	or
	`RP/0/RP0/CPU0:router(config-ospf-ar-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 9	**show ospf** [*process-name*] [*area-id*] **mpls traffic-eng** {**link** | **fragment**}
	Example:
	`RP/0/RP0/CPU0:router# show ospf 1 0 mpls traffic-eng link`
		(Optional) Displays information about the links and fragments available on the local router for MPLS TE.

Examples

This section provides the following output examples:

- Sample Output for the show ospf Command Before Configuring MPLS TE, page RC-178
- Sample Output for the show ospf mpls traffic-eng Command, page RC-179
- Sample Output for the show ospf Command After Configuring MPLS TE, page RC-180

Sample Output for the show ospf Command Before Configuring MPLS TE

In the following example, the **show route ospf** EXEC command verifies that POS interface 0/3/0/0 exists and MPLS TE is not configured:

```
RP/0/RP0/CPU0:router# show route ospf 1 0

O E2 192.168.10.0/24 [110/20] via 192.168.1.2, 00:02:50, POS 0/3/0/0
                [110/20] via 192.168.4.1, 00:02:50, POS 0/3/0/1
O E2 192.168.11.0/24 [110/20] via 192.168.1.2, 00:02:50, POS 0/3/0/0
                [110/20] via 192.168.4.1, 00:02:50, POS 0/3/0/1
O E2 192.168.244.0/24 [110/20] via 192.168.1.2, 00:02:50, POS 0/3/0/0
                   [110/20] via 192.168.4.1, 00:02:50, POS 0/3/0/1
O    192.168.12.0/24 [110/2] via 192.168.1.2, 00:02:50, POS 0/3/0/0
                    [110/2] via 192.168.4.1, 00:02:50, POS 0/3/0/1
```

Sample Output for the show ospf mpls traffic-eng Command

In the following example, the **show ospf mpls traffic-eng** EXEC command verifies that the MPLS TE fragments are configured correctly:

```
RP/0/RP0/CPU0:router# show ospf 1 mpls traffic-eng fragment

OSPF Router with ID (192.168.4.3) (Process ID 1)

  Area 0 has 1  MPLS TE fragment. Area instance is 3.
  MPLS router address is 192.168.4.2
  Next fragment ID is 1

  Fragment 0 has 1 link. Fragment instance is 3.
  Fragment has 0 link the same as last update.
  Fragment advertise MPLS router address
    Link is associated with fragment 0. Link instance is 3
      Link connected to Point-to-Point network
      Link ID :55.55.55.55
      Interface Address :192.168.50.21
      Neighbor Address :192.168.4.1
      Admin Metric :0
      Maximum bandwidth :19440000
      Maximum global pool reservable bandwidth :25000000
      Maximum sub pool reservable bandwidth    :3125000
      Number of Priority :8
      Global pool unreserved BW
      Priority 0 :  25000000  Priority 1 :  25000000
      Priority 2 :  25000000  Priority 3 :  25000000
      Priority 4 :  25000000  Priority 5 :  25000000
      Priority 6 :  25000000  Priority 7 :  25000000
      Sub pool unreserved BW
      Priority 0 :   3125000  Priority 1 :   3125000
      Priority 2 :   3125000  Priority 3 :   3125000
      Priority 4 :   3125000  Priority 5 :   3125000
      Priority 6 :   3125000  Priority 7 :   3125000
      Affinity Bit :0
```

In the following example, the **show ospf mpls traffic-eng** EXEC command verifies that the MPLS TE links on area instance 3 are configured correctly:

```
RP/0/RP0/CPU0:router# show ospf mpls traffic-eng link

            OSPF Router with ID (192.168.4.1) (Process ID 1)

  Area 0 has 1  MPLS TE links. Area instance is 3.

  Links in hash bucket 53.
    Link is associated with fragment 0. Link instance is 3
      Link connected to Point-to-Point network
      Link ID :192.168.50.20
      Interface Address :192.168.20.50
      Neighbor Address :192.168.4.1
      Admin Metric :0
      Maximum bandwidth :19440000
      Maximum global pool reservable bandwidth :25000000
      Maximum sub pool reservable bandwidth    :3125000
      Number of Priority :8
      Global pool unreserved BW
      Priority 0 :  25000000  Priority 1 :  25000000
      Priority 2 :  25000000  Priority 3 :  25000000
      Priority 4 :  25000000  Priority 5 :  25000000
      Priority 6 :  25000000  Priority 7 :  25000000
      Sub pool unreserved BW
```

```
Priority 0 :   3125000  Priority 1 :   3125000
Priority 2 :   3125000  Priority 3 :   3125000
Priority 4 :   3125000  Priority 5 :   3125000
Priority 6 :   3125000  Priority 7 :   3125000
Affinity Bit :0
```

Sample Output for the show ospf Command After Configuring MPLS TE

In the following example, the **show route ospf** EXEC command verifies that the MPLS TE tunnels replaced POS interface 0/3/0/0 and that configuration was performed correctly:

```
RP/0/RP0/CPU0:router# show route ospf 1 0

O E2 192.168.10.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O E2 192.168.11.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O E2 192.168.1244.0/24 [110/20] via 0.0.0.0, 00:00:15, tunnel2
O    192.168.12.0/24 [110/2] via 0.0.0.0, 00:00:15, tunnel2
```

Verifying OSPF Configuration and Operation

This task explains how to verify the configuration and operation of OSPF.

Note To execute OSPFv3 commands for this task, replace **ospf** with **ospfv3** in Steps 1 through 7.

SUMMARY STEPS

1. **show ospf** [*process-name*]
2. **show ospf** [*process-name*] **border-routers** [*router-id*]
3. **show ospf** [*process-name*] **database**
4. **show ospf** [*process-name*] [*area-id*] **flood-list interface** *type instance*
5. **show ospf** [*process-name*] [*area-id*] **neighbor** [*interface-type interface-instance*] [*neighbor-id*] [**detail**]
6. **clear ospf** [*process-name*] **process**
7. **clear ospf** [*process-name*] **statistics** [**neighbor** [*interface-type interface-instance*] [*ip-address*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show ospf** [*process-name*] **Example:** RP/0/RP0/CPU0:router# show ospf group1	(Optional) Displays general information about OSPF routing processes.
Step 2	**show ospf** [*process-name*] **border-routers** [*router-id*] **Example:** RP/0/RP0/CPU0:router# show ospf group1 border-routers	(Optional) Displays the internal OSPF routing table entries to an ABR and ASBR.

	Command or Action	Purpose
Step 3	**show ospf** [*process-name*] **database** **Example:** `RP/0/RP0/CPU0:router# show ospf group2 database`	(Optional) Displays the lists of information related to the OSPF database for a specific router. • The various forms of this command deliver information about different OSPF LSAs.
Step 4	**show ospf** [*process-name*] [*area-id*] **flood-list interface** *type instance* **Example:** `RP/0/RP0/CPU0:router# show ospf 100 flood-list interface pos 0/3/0/0`	(Optional) Displays a list of OSPF LSAs waiting to be flooded over an interface.
Step 5	**show ospf** [*process-name*] [*area-id*] **neighbor** [*interface-type interface-instance*] [*neighbor-id*] [**detail**] **Example:** `RP/0/RP0/CPU0:router# show ospf 100 neighbor`	(Optional) Displays OSPF neighbor information on an individual interface basis.
Step 6	**clear ospf** [*process-name*] **process** **Example:** `RP/0/RP0/CPU0:router# clear ospf 100 process`	(Optional) Resets an OSPF router process without stopping and restarting it.
Step 7	**clear ospf** [*process-name*] **statistics** [**neighbor** [*interface-type interface-instance*] [*ip-address*]] **Example:** `RP/0/RP0/CPU0:router# clear ospf 100 statistics`	(Optional) Clears the OSPF statistics of neighbor state transitions.

Configuring OSPFv3 Graceful Restart

This section describes the following tasks for configuring a graceful restart of an OSPFv3 process:

- Enabling Graceful Restart, page RC-182
- Configuring the Maximum Lifetime of a Graceful Restart, page RC-182
- Configuring the Minimum Time Required Between Restarts, page RC-183
- Configuring the Helper Level of the Router, page RC-184
- Displaying Information About Graceful Restart, page RC-185

Enabling Graceful Restart

This section describes how to enable an OSPFv3 graceful restart on the current router. By default, this feature is disabled.

SUMMARY STEPS

1. **configuration**
2. **router ospfv3**
3. **graceful-restart**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`config`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr#config`
	`RP/0/RP0/CPU0:single10-hfr(config)`
		Enters global configuration mode.

Step 2	**`router ospfv3`** *`process-name`*
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config)# router ospfv3 test`
		Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.

Step 3	**`graceful-restart`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config-ospfv3)#graceful-restart`
		Enable graceful restart on the current router.

Configuring the Maximum Lifetime of a Graceful Restart

This section describes the task of modifying the total time that a router can be in graceful restart mode. The default lifetime is 95 seconds. The range is 90–3600 seconds.

SUMMARY STEPS

1. **configuration**
2. **router ospfv3**
3. **graceful-restart lifetime**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`config`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr#config`
	`RP/0/RP0/CPU0:single10-hfr(config)`
		Enters global configuration mode.

Step 2	**`router ospfv3`** *`<process-name>`*
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config)# router ospfv3 test`
		Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.

Step 3	**`graceful-restart lifetime`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config-ospfv3)#graceful-restart lifetime 120`
		Specifies a maximum duration for a graceful restart.

Configuring the Minimum Time Required Between Restarts

This section describes the task of modifying the minimal time that is required between allowable graceful restarts. The purpose of this interval is to prevent the waste of system resources if the OSPFv3 process is repeatedly crashing for reasons that must be diagnosed. The default value for the interval is 90 seconds. The range is 90–3600 seconds.

SUMMARY STEPS

1. **configuration**
2. **router ospfv3**
3. **graceful-restart interval**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**config**
	Example:
	`RP/0/RP0/CPU0:single10-hfr#config`
	`RP/0/RP0/CPU0:single10-hfr(config)`
		Enters global configuration mode.

Step 2	**router ospfv3** *<process-name>* **Example:** `RP/0/RP0/CPU0:single10-hfr(config)# router ospfv3 test`	Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.
Step 3	**graceful-restart interval** *<seconds>* **Example:** `RP/0/RP0/CPU0:single10-hfr(config-ospfv3)#graceful-restart interval 120`	Specifies the interval (minimal time) between graceful restarts on the current router.

Configuring the Helper Level of the Router

This section describes the task of disabling the helper mode on the current router. By default, a router that is capable of doing an OSPFv3 graceful restart is also enabled to be a helper to a node in graceful mode. The **graceful-restart helper** command lets you disable the current router's helper capability.

SUMMARY STEPS

1. **configuration**
2. **router ospfv3**
3. **graceful-restart helper** [**disable**]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`config`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr#config`
	`RP/0/RP0/CPU0:single10-hfr(config)`
		Enters global configuration mode.

Step 2	**`router ospfv3`** *`<process-name>`*
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config)# router ospfv3 test`
		Enters router configuration mode for OSPFv3. The process name is a WORD that uniquely identifies an OSPF routing process. The process name is any alphanumeric string no longer than 40 characters without spaces.

Step 3	**`graceful-restart helper`**
	Example:
	`RP/0/RP0/CPU0:single10-hfr(config-ospfv3)#graceful-restart helper disable`
		Disables the helper capability.

Displaying Information About Graceful Restart

This section describes the tasks you can use to display information about a graceful restart.

- To see if the feature is enabled and when the last graceful restart ran, use the **show ospf** command. To see details for an OSPFv3 instance, use the **show ospf** *process-name* **database grace** command.

Displaying the State of the Graceful Restart Feature

The following screen output shows the state of the graceful restart capability on the local router:

```
RP/0/0/CPU0:LA#show ospfv3 test database grace

 Routing Process “ospfv3 test” with ID 2.2.2.2
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Initial LSA throttle delay 0 msecs
 Minimum hold time for LSA throttle 5000 msecs
 Maximum wait time for LSA throttle 5000 msecs
 Minimum LSA arrival 1000 msecs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Maximum number of configured interfaces 255
 Number of external LSA 0. Checksum Sum 00000000
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
 Graceful Restart enabled, last GR 11:12:26 ago (took 6 secs)
    Area BACKBONE(0)
        Number of interfaces in this area is 1
        SPF algorithm executed 1 times
        Number of LSA 6. Checksum Sum 0x0268a7
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
RP/0/0/CPU0:LA#
```

Displaying Graceful Restart Information for an OSPFv3 Instance

The following screen output shows the link state for the instance of OSPFv3 called test:

```
RP/0/0/CPU0:LA#show ospfv3 test database grace

            OSPFv3 Router with ID (2.2.2.2) (Process ID test)

            Router Link States (Area 0)
ADV Router      Age         Seq#             Fragment ID  Link count  Bits
1.1.1.1               1949        0x8000000e      0                      1            None
2.2.2.2               2007        0x80000011      0                      1            None

            Link (Type-8) Link States (Area 0)
ADV Router      Age         Seq#              Link ID    Interface
1.1.1.1               180          0x80000006    1            PO0/2/0/0
s2.2.2.2             2007        0x80000006    1            PO0/2/0/0

            Intra Area Prefix Link States (Area 0)
ADV Router      Age         Seq#              Link ID    Ref-lstype  Ref-LSID
1.1.1.1               180         0x80000006    0               0x2001          0
2.2.2.2               2007        0x80000006   0               0x2001          0

            Grace (Type-11) Link States (Area 0)
ADV Router      Age         Seq#                Link ID    Interface
2.2.2.2                2007        0x80000005    1              PO0/2/0/0

RP/0/0/CPU0:LA#
```

Enabling Multicast-Intact for OSPFv2

This optional task describes how to enable multicast-intact for OSPFv2 routes that use IPv4 addresses.

Summary Steps

1. **configure**
2. **router ospf** *instance-id*
3. **mpls traffic-eng multicast-intact**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`router ospf` *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf isp`	Enables OSPF routing for the specified routing process, and places the router in router configuration mode. In this example, the OSPF instance is called isp.

	Command or Action	Purpose
Step 3	**mpls traffic-eng multicast-intact** **Example:** `RP/0/RP0/CPU0:router(config-isis)# mpls traffic-eng multicast-intact`	Enables multicast-intact.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-isis-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing OSPF on Cisco IOS XR Software

This section provides the following configuration examples:

- Cisco IOS XR for OSPF Version 2 Configuration: Example, page RC-188
- CLI Inheritance and Precedence for OSPF Version 2: Example, page RC-189
- MPLS TE for OSPF Version 2: Example, page RC-190
- ABR with Summarization for OSPFv3: Example, page RC-190
- ABR Stub Area for OSPFv3: Example, page RC-190
- ABR Totally Stub Area for OSPFv3: Example, page RC-191
- Route Redistribution for OSPFv3: Example, page RC-191
- Virtual Link Configured Through Area 1 for OSPFv3: Example, page RC-191

Cisco IOS XR for OSPF Version 2 Configuration: Example

The following example shows how an OSPF interface is configured for an area in Cisco IOS XR software.

In Cisco IOS XR software, area 0 must be explicitly configured with the **area** command and all interfaces that are in the range from 10.1.2.0 to 10.1.2.255 are bound to area 0. Interfaces are configured with the **interface** command (while the router is in area configuration mode) and the **area** keyword is not included in the interface statement.

Cisco IOS XR Software Configuration

```
interface POS 0/3/0/0
 ip address 10.1.2.1 255.255.255.255
 negotiation auto
!
router ospf 1
router-id 10.2.3.4
 area 0
  interface POS 0/3/0/0
!
!
```

The following example shows how OSPF interface parameters are configured for an area in Cisco IOS XR software.

In Cisco IOS XR software, OSPF interface-specific parameters are configured in interface configuration mode and explicitly defined for area 0. In addition, the **ip ospf** keywords are no longer required.

Cisco IOS XR Software Configuration

```
interface POS 0/3/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
!
router ospf 1
 router-id 10.2.3.4
 area 0
  interface POS 0/3/0/0
   cost 77
   mtu-ignore
   authentication message-digest
   message-digest-key 1 md5 0 test
!
!
```

The following example shows the hierarchical CLI structure of Cisco IOS XR software.

In Cisco IOS XR software, OSPF areas must be explicitly configured, and interfaces configured under the area configuration mode are explicitly bound to that area. In this example, interface 10.1.2.0/24 is bound to area 0 and interface 10.1.3.0/24 is bound to area 1.

Cisco IOS XR Software Configuration

```
interface POS 0/3/0/0
 ip address 10.1.2.1 255.255.255.0
 negotiation auto
!
interface POS 0/3/0/1
 ip address 10.1.3.1 255.255.255.0
 negotiation auto
!
```

```
router ospf 1
 router-id 10.2.3.4
 area 0
  interface POS 0/3/0/0
!
 area 1
  interface POS 0/3/0/1
!
!
```

CLI Inheritance and Precedence for OSPF Version 2: Example

The following example configures the cost parameter at different hierarchical levels of the OSPF topology, and illustrates how the parameter is inherited and how only one setting takes precedence. According to the precedence rule, the most explicit configuration is used.

The cost parameter is set to 5 in router configuration mode for the OSPF process. Area 1 sets the cost to 15 and area 6 sets the cost to 30. All interfaces in area 0 inherit a cost of 5 from the OSPF process because the cost was not set in area 0 or its interfaces.

In area 1, every interface has a cost of 15 because the cost is set in area 1 and 15 overrides the value 5 that was set in router configuration mode.

Area 4 does not set the cost, but POS interface 01/0/2 sets the cost to 20. The remaining interfaces in area 4 have a cost of 5 that is inherited from the OSPF process.

Area 6 sets the cost to 30, which is inherited by POS interfaces 0/1/0/3 and 0/2/0/3. POS interface 0/3/0/3 uses the cost of 1, which is set in interface configuration mode.

```
router ospf 1
 router-id 10.5.4.3
 cost 5
 area 0
  interface POS 0/1/0/0
  !
  interface POS 0/2/0/0
  !
  interface POS 0/3/0/0
  !
 !
 area 1
  cost 15
  interface POS 0/1/0/1
  !
  interface POS 0/2/0/1
  !
  interface POS 0/3/0/1
  !
 !
 area 4
  interface POS 0/1/0/2
   cost 20
  !
  interface POS 0/2/0/2
  !
  interface POS 0/3/0/2
  !
 !
 area 6
  cost 30
  interface POS 0/1/0/3
  !
  interface POS 0/2/0/3
  !
  interface POS 0/3/0/3
   cost 1
  !
 !
```

MPLS TE for OSPF Version 2: Example

The following example shows how to configure the OSPF portion of MPLS TE. However, you still need to build an MPLS TE topology and create an MPLS TE tunnel. See the *Cisco IOS XR MPLS Configuration Guide* for information.

In this example, loopback interface 0 is associated with area 0 and area 0 is declared to be an MPLS area:

```
interface Loopback 0
 ip address 10.10.10.10 255.255.255.0
!
interface POS 0/2/0/0
 ip address 10.1.2.2 255.255.255.0
!
router ospf 1
 router-id 10.10.10.10
 nsf
 auto-cost reference-bandwidth 10000
 area 0
  interface POS 0/2/0/0
  interface Loopback 0
 mpls traffic-eng area 0
 mpls traffic-eng router-id Loopback 0
```

ABR with Summarization for OSPFv3: Example

The following example shows the prefix range 2300::/16 summarized from area 1 into the backbone:

```
router ospfv3 1
 router-id 192.168.0.217
 area 0
  interface POS 0/2/0/1
 area 1
  range 2300::/16
  interface POS 0/2/0/0
```

ABR Stub Area for OSPFv3: Example

The following example shows that area 1 is configured as a stub area:

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface POS 0/2/0/1
 area 1
  stub
  interface POS 0/2/0/0
```

ABR Totally Stub Area for OSPFv3: Example

The following example shows that area 1 is configured as a totally stub area:

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface POS 0/2/0/1
 area 1
  stub no-summary
  interface POS 0/2/0/0
```

Route Redistribution for OSPFv3: Example

The following example uses prefix lists to limit the routes redistributed from other protocols.

Only routes with 9898:1000 in the upper 32 bits and with prefix lengths from 32 to 64 are redistributed from BGP 42. Only routes *not* matching this pattern are redistributed from BGP 1956.

```
ipv6 prefix-list list1
 seq 10 permit 9898:1000::/32 ge 32 le 64

ipv6 prefix-list list2
 seq 10 deny 9898:1000::/32 ge 32 le 64
 seq 20 permit ::/0 le 128

router ospfv3 1
 router-id 10.0.0.217
 redistribute bgp 42
 redistribute bgp 1956
 distribute-list prefix-list list1 out bgp 42
 distribute-list prefix-list list2 out bgp 1956
 area 1
  interface POS 0/2/0/0
```

Virtual Link Configured Through Area 1 for OSPFv3: Example

This example shows how to set up a virtual link to connect the backbone through area 1 for the OSPFv3 topology that consists of areas 0 and 1 and virtual links 10.0.0.217 and 10.0.0.212:

ABR 1 Configuration

```
router ospfv3 1
 router-id 10.0.0.217
 area 0
  interface POS 0/2/0/1
 area 1
  virtual-link 10.0.0.212
  interface POS 0/2/0/0
```

ABR 2 Configuration

```
router ospfv3 1
 router-id 10.0.0.212
 area 0
  interface POS 0/3/0/1
 area 1
  virtual-link 10.0.0.217
  interface POS 0/2/0/0
```

Virtual Link Configured with MD5 Authentication for OSPF Version 2: Example

The following examples show how to configure a virtual link to your backbone and apply MD5 authentication. You must perform the steps described on both ABRs at each end of the virtual link.

After you explicitly configure the ABRs, the configuration is inherited by all interfaces bound to that area—unless you override the values and configure them explicitly for the interface.

To understand virtual links, see the "Virtual Link and Transit Area for OSPF" section on page RC-138.

In this example, all interfaces on router ABR1 use MD5 authentication:

```
router ospf ABR1
  router-id 10.10.10.10
  authentication message-digest
  message-digest-key 100 md5 0 cisco
  area 0
    interface pos 0/2/0/1
    interface pos 0/3/0/0
  area 1
    interface pos 0/3/0/1
    virtual-link 10.10.5.5
  !
!
```

In this example, only area 1 interfaces on router ABR3 use MD5 authentication:

```
router ospf ABR2
  router-id 10.10.5.5
  area 0
  area 1
    authentication message-digest
    message-digest-key 100 md5 0 cisco
    interface pos 0/9/0/1
    virtual-link 10.10.10.10
  area 3
    interface Loopback 0
    interface pos 0/9/0/0
  !
!
```

Where to Go Next

To configure route maps through the RPL for OSPF Version 2, see the *Implementing Routing Policy on Cisco IOS XR Software* document.

To build an MPLS TE topology, create tunnels, and configure forwarding over the tunnel for OSPF Version 2; see the *Cisco IOS XR MPLS Configuration Guide*.

Additional References

The following sections provide references related to implementing OSPF on Cisco IOS XR software.

Related Documents

Related Topic	Document Title
OSPF and OSPFv3 commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*, Release 32
MPLS TE feature information	*Implementing MPLS Traffic Engineering on Cisco IOS XR Software* module in the *Cisco IOS XR MPLS Configuration Guide*, Release 3.2

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
• OSPF-MIB	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1587	*Not so Stubby Area (NSSA)*
RFC 1793	*OSPF over demand circuit*
RFC 2328	*OSPF Version 2*
RFC 2740	*OSPFv3*
RFC 3623	*Graceful OSPF Restart (OSPFv2)*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing and Monitoring RIB on Cisco IOS XR Software

Routing Information Base (RIB) is a distributed collection of information about routing connectivity among all nodes of a network.

Each router maintains a RIB containing the routing information for that router. RIB stores the best routes from all routing protocols that are running on the system.

This module describes the tasks you need to perform to implement and monitor RIB on your Cisco IOS XR network.

Note For more information about RIB on the Cisco IOS XR software and complete descriptions of RIB commands listed in this module, see the "Related Documents" of this module. To locate documentation for other commands that might appear during the execution of a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing and Monitoring RIB on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.

Contents

Prerequisites for Implementing RIB on Cisco IOS XR Software

- To use this command, you must be in a user group associated with a task group that includes the proper task IDs. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- RIB is distributed with the base Cisco IOS XR software; as such, it does not have any special requirements for installation. The following are the requirements for base software installation:
  - Router
  - Cisco IOS XR software
  - Base package

Information About RIB Configuration

To implement the Cisco RIB feature, you must understand the following concepts:

- Overview of RIB, page RC-196
- RIB Data Structures in BGP and Other Protocols, page RC-196
- RIB Administrative Distance, page RC-197
- RIB Support for IPv4 and IPv6, page RC-197

Overview of RIB

Each routing protocol selects its own set of best routes and installs those routes and their attributes in RIB. RIB stores these routes and selects the best ones from among all routing protocols. Those routes are downloaded to the line cards for use in forwarding packets. The acronym RIB is used both to refer to RIB processes and the collection of route data contained within RIB.

Within a protocol, routes are selected based on the metrics in use by that protocol. A protocol downloads its best routes (lowest or tied metric) to RIB. RIB selects the best overall route by comparing the administrative distance of the associated protocol.

RIB Data Structures in BGP and Other Protocols

RIB uses processes and maintains data structures distinct from other routing applications, such as Border Gateway Protocol (BGP) and other unicast routing protocols, or multicast protocols, such as Protocol Independent Multicast (PIM) or Multicast Source Discovery Protocol (MSDP). However, these routing protocols use internal data structures similar to what RIB uses, and may internally refer to the data structures as a RIB. For example, BGP routes are stored in the BGP RIB (BRIB), and multicast routes, computed by multicast routing protocols such as PIM and MSDP, are stored in the Multicast RIB (MRIB). RIB processes are not responsible for the BRIB and MRIB, which are handled by BGP and multicast processes, respectively.

The table used by the line cards and RP to forward packets is called the Forwarding Information Base (FIB). RIB processes do not build the FIBs. Instead, RIB downloads the set of selected best routes to the FIB processes, by the Bulk Content Downloader (BCDL) process, onto each line card. FIBs are then constructed.

RIB Administrative Distance

Forwarding is done based on the longest prefix match. If you are forwarding a packet destined to 10.0.2.1, you prefer 10.0.2.0/24 over 10.0.0.0/16 because the mask /24 is longer (and more specific) than a /16.

Routes from different protocols that have the same prefix and length are chosen based on administrative distance. For instance, the Open Shortest Path First (OSPF) protocol has an administrative distance of 110, and the Intermediate System-to-Intermediate System (IS-IS) protocol has an administrative distance of 115. If IS-IS and OSPF both download 10.0.1.0/24 to RIB, RIB would prefer the OSPF route because OSPF has a lower administrative distance. Administrative distance is used only to choose between multiple routes of the same length.

The default administrative distances for the common protocols are shown in Table 2.

Table 2 Default Administrative Distances

Protocol	Administrative Distance Default
Connected or local routes	0
Static routes	1
External BGP routes	20
OSPF routes	110
IS-IS routes	115
Internal BGP routes	200

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.

Note Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

RIB Support for IPv4 and IPv6

In Cisco IOS XR software, RIB tables support multicast and unicast routing.

The default routing table for Cisco IOS XR RIB are the unicast and the multicast-unicast RIB tables for IPv4 and IPv6 routing, respectively. For multicast routing, routing protocols insert unicast routes into the multicast-unicast RIB table. Multicast protocols then use the information to build multicast routes (which in turn are stored in the MRIB). See the multicast documentation for more information on using and configuring multicast.

RIB processes ipv4_rib and ipv6_rib run on the RP card. If process placement functionality is available and supported by multiple RPs in the router, RIB processes can be placed on any available node.

How to Deploy and Monitor RIB

To deploy and monitor RIB, you must understand the following concepts:

- Verifying RIB Configuration Using the Routing Table, page RC-198 (required)
- Verifying Networking and Routing Problems, page RC-198 (required)

Verifying RIB Configuration Using the Routing Table

This task verifies the RIB configuration to ensure that RIB is running on the RP and functioning properly by checking the routing table summary and details.

SUMMARY STEPS

1. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **summary**
2. **show route** [**protocol** [*process-id*]] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] [*ip-address* [*mask*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **summary** **Example:** `RP/0/RP0/CPU0:router# show route summary`	Displays route summary information on the specified routing table. • The default table summarized is the IPv4 unicast routing table.
Step 2	**show route** [**protocol** [*process-id*]] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] [*ip-address* [*mask*]] **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast`	Displays more detailed route information on the specified routing table. • This command is usually issued with an IP address or other optional filters to limit its display. Otherwise, it displays all routes from the default IPv4 unicast routing table, which can result in an extensive list, depending on the configuration of the network.

Verifying Networking and Routing Problems

This task verifies the operation of the routes between nodes.

SUMMARY STEPS

1. **show route** [*protocol* [*instance*]] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] [*ip-address* [*mask*]]
2. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **backup** [*ip-address*]
3. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **best-local** *ip-address*
4. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **connected**

5. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **local** [*interface*]
6. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] *ip-address mask* ***longer-prefixes***
7. **show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **next-hop** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show route** [*protocol* [*instance*]] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] [*ip-address* [*mask*]] **Example:** `RP/0/RP0/CPU0:router# show route list list1 bgp aspo ipv4 unicast 192.168.111/8`	Displays the current routes in RIB.
Step 2	**show route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **backup** [*ip-address*] **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast backup 192.168.111/8`	Displays backup routes in RIB.
Step 3	**sho**w **route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] best-l**ocal** *ip-address* **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast best-local 192.168.111/8`	Displays the best-local address to use for return packets from the given destination.
Step 4	**sho**w **route** [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **connected** **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast connected`	Displays the current connected routes of the routing table.
Step 5	**show rout**e [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **local** [*interface*] **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast local`	Displays local routes for receive entries in the routing table.

	Command or Action	Purpose
Step 6	**show rout**e [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] ip-address *mask* **longer-prefixes** **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast 192.168.111/8 longer-prefixes`	Displays the current routes in RIB that share a given number of bits with a given network.
Step 7	**show rout**e [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **multicast** | **safi-all**] **next-hop** *ip-address* **Example:** `RP/0/RP0/CPU0:router# show route ipv4 unicast next-hop 192.168.1.34`	Displays the next hop gateway or host to a destination address.

Configuration Examples for RIB Monitoring

RIB is not configured separately for the Cisco IOS XR system. RIB computes connectivity of the router with other nodes in the network based on input from the routing protocols. RIB may be used to monitor and troubleshoot the connections between RIB and its clients, but it is essentially used to monitor routing connectivity between the nodes in a network. This section contains displays from the **show** commands used to monitor that activity. The following sample output is provided:

- Output of show route Command: Example, page RC-200
- Output of show route backup Command: Example, page RC-201
- Output of show route best-local Command: Example, page RC-201
- Output of show route connected Command: Example, page RC-201
- Output of show route local Command: Example, page RC-201
- Output of show route longer-prefixes Command: Example, page RC-202
- Output of show route next-hop Command: Example, page RC-202

Output of show route Command: Example

The following is sample output from the **show route** command when entered without an address:

```
RP/0/RP0/CPU0:router# show route

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       O - OSPF, IA - OSPF inter area, N1 - OSPF NSSA external type 1
       N2 - OSPF NSSA external type 2, E1 - OSPF external type 1
       E2 - OSPF external type 2, E - EGP, i - ISIS, L1 - IS-IS level-1
       L2 - IS-IS level-2, ia - IS-IS inter area
       su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is 172.23.54.1 to network 0.0.0.0

C    10.2.210.0/24 is directly connected, 1d21h, Ethernet0/1/0/0
L    10.2.210.221/32 is directly connected, 1d21h, Ethernet0/1/1/0
C    172.20.16.0/24 is directly connected, 1d21h, ATM4/0.1
```

```
L    172.20.16.1/32 is directly connected, 1d21h, ATM4/0.1
C    10.6.100.0/24 is directly connected, 1d21h, Loopback1
L    10.6.200.21/32 is directly connected, 1d21h, Loopback0
S    192.168.40.0/24 [1/0] via 172.20.16.6, 1d21h
```

Output of show route backup Command: Example

The following is sample output from the **show route backup** command:

```
RP/0/RP0/CPU0:router# show route backup

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       O - OSPF, IA - OSPF inter area, N1 - OSPF NSSA external type 1
       N2 - OSPF NSSA external type 2, E1 - OSPF external type 1
       E2 - OSPF external type 2, E - EGP, i - ISIS, L1 - IS-IS level-1
       L2 - IS-IS level-2, ia - IS-IS inter area
       su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

S    172.73.51.0/24 is directly connected, 2d20h, GigabitEthernet2/2
                 Backup  O E2 [110/1] via 10.12.12.2, POS3/0
```

Output of show route best-local Command: Example

The following is sample output from the **show route best-local** command:

```
RP/0/RP0/CPU0:router# show route best-local 10.12.12.1

Routing entry for 10.12.12.1/32
  Known via "local", distance 0, metric 0 (connected)
  Routing Descriptor Blocks
    10.12.12.1 directly connected, via POS3/0
      Route metric is 0
```

Output of show route connected Command: Example

The following is sample output from the **show route connected** command:

```
RP/0/RP0/CPU0:router# show route connected

Gateway of last resort is 172.23.54.1 to network 0.0.0.0

C    10.2.210.0/24 is directly connected, 1d21h, Ethernet0
C    172.20.16.0/24 is directly connected, 1d21h, ATM4/0.1
C    10.6.100.0/24 is directly connected, 1d21h, Loopback1
```

Output of show route local Command: Example

The following is sample output from the **show route local** command:

```
RP/0/RP0/CPU0:router# show route local

L    10.10.10.1/32 is directly connected, 00:14:36, Loopback0
L    10.91.36.98/32 is directly connected, 00:14:32, Ethernet0/0
L    172.22.12.1/32 is directly connected, 00:13:35, POS3/0
```

```
L    192.168.20.2/32 is directly connected, 00:13:27, GigabitEthernet2/0
L    10.254.254.1/32 is directly connected, 00:13:26, GigabitEthernet2/2
```

Output of show route longer-prefixes Command: Example

The following is sample output from the **show route longer-prefixes** command:

```
RP/0/RP0/CPU0:router# show route ipv4 172.16.0.0/8 longer-prefixes

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       O - OSPF, IA - OSPF inter area, N1 - OSPF NSSA external type 1
       N2 - OSPF NSSA external type 2, E1 - OSPF external type 1
       E2 - OSPF external type 2, E - EGP, i - ISIS, L1 - IS-IS level-1
       L2 - IS-IS level-2, ia - IS-IS inter area
       su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is 172.23.54.1 to network 0.0.0.0
S    172.16.2.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.3.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.4.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.5.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.6.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.7.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.8.0/32 is directly connected, 00:00:24, Loopback0
S    172.16.9.0/32 is directly connected, 00:00:24, Loopback0
```

Output of show route next-hop Command: Example

The following is sample output from the **show route next-hop** command:

```
RP/0/RP0/CPU0:router# show route next-hop 10.0.0.1

Routing entry for 10.0.0.0/24
  Known via "connected", distance 0, metric 0 (connected)
  Routing Descriptor Blocks
    10.0.0.50 directly connected, via GigabitEthernet6/0
      Route metric is 0
```

Where to Go Next

For additional information on the protocols that interact with RIB, you may want to see the following publications:

- *Implementing BGP on Cisco IOS XR Software*
- *Implementing IS-IS on Cisco IOS XR Software*
- *Implementing OSPF on Cisco IOS XR Software*
- *RIB Commands on Cisco IOS XR Software*

Additional References

The following sections provide references related to implementing RIB on Cisco IOS XR software:

Related Documents

Related Topic	Document Title
Routing Information Base commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*RIB Commands on Cisco IOS XR Software* in the *Cisco IOS XR Routing Command Reference*, Release 3.2
BGP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*BGP Commands on Cisco IOS XR Software*, in the *Cisco IOS XR Routing Command Reference*, Release 3.2
IS-IS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IS-IS Commands on Cisco IOS XR Software* in the *Cisco IOS XR Routing Command Reference*, Release 3.2
OSPF commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*OSPF Commands on Cisco IOS XR Software* in the *Cisco IOS XR Routing Command Reference*, Release 3.2
OSPFv3 commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*OSPFv3 Commands on Cisco IOS XR Software* in the *Cisco IOS XR Routing Command Reference*, Release 3.2
Multicast commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Multicast Command Reference*, Release 3.2
Multicast configuration: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Multicast Configuration Guide*, Release 3.2

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
• IP-FORWARD-MIB • RFC1213-MIB	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Routing Policy on Cisco IOS XR Software

A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another. Routing protocols make decisions to advertise, aggregate, discard, distribute, export, hold, import, redistribute and otherwise modify routes based on configured routing policy.

The routing policy language (RPL) has been designed to provide a single, straightforward language in which all routing policy needs can be expressed. RPL was designed to support large-scale routing configurations. It greatly reduces the redundancy inherent in previous routing policy configuration methods. RPL has been designed to streamline routing policy configuration, to reduce system resources required to store and process these configurations, and to simplify troubleshooting.

Note For more information about routing policy on the Cisco IOS XR software and complete descriptions of the routing policy commands listed in this module, see the "Related Documents" section of this module. To locate documentation for other commands that might appear during execution of a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing Routing Policy on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.

Contents

Prerequisites for Implementing Routing Policy

The following are prerequisites for implementing Routing Policy on Cisco IOS XR Software:

- To use this command, you must be in a user group associated with a task group that includes the proper task IDs. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.
- Border Gateway Protocol (BGP), integrated Intermediate System-to-Intermediate System (IS-IS), or Open Shortest Path First (OSPF) must be configured in your network.

Information About Implementing Routing Policy

To implement RPL, you need to understand the following concepts:

- Routing Policy Language, page RC-206
- Routing Policy Language, page RC-206
- Routing Policy Configuration Basics, page RC-213
- Policy Definitions, page RC-213
- Parameterization, page RC-214
- Semantics of Policy Application, page RC-215
- Policy Statements, page RC-219
- Attach Points, page RC-223
- Attached Policy Modification, page RC-235
- Nonattached Policy Modification, page RC-235

Routing Policy Language

This section contains the following information:

- Routing Policy Language Overview, page RC-206
- Routing Policy Language Structure, page RC-207
- Routing Policy Language Components, page RC-211
- Routing Policy Language Usage, page RC-211

Routing Policy Language Overview

RPL was developed to support large-scale routing configurations. RPL has several fundamental capabilities that differ from those present in configurations oriented to traditional route maps, access lists, and prefix lists. The first of these capabilities is the ability to build policies in a modular form. Common blocks of policy can be defined and maintained independently. These common blocks of policy can then be applied from other blocks of policy to build complete policies. This capability reduces the amount of configuration information that needs to be maintained. In addition, these common blocks of policy can be parameterized. This parameterization allows for policies that share the same structure but differ in the specific values that are set or matched against to be maintained as independent blocks of policy. For example, three policies that are identical in every way except for the local preference value they set can be represented as one common parameterized policy that takes the varying local preference value as a parameter to the policy.

The policy language introduces the notion of sets. Sets are containers of similar data that can be used in route attribute matching and setting operations. Four set types exist: prefix-sets, community-sets, as-path-sets, and extcommunity-sets. These sets hold groupings of IPv4 or IPv6 prefixes, community values, AS path regular expressions, and extended community values, respectively. Sets are simply containers of data. Most sets also have an inline variant. An inline set allows for small enumerations of values to be used directly in a policy rather than having to refer to a named set. Prefix lists, community lists, and AS path lists must be maintained even when only one or two items are in the list. An inline set in RPL allows the user to place small sets of values directly in the policy body without having to refer to a named set.

Decision making, such as accept and deny, is explicitly controlled by the policy definitions themselves. RPL combines matching operators, which may use set data, with the traditional Boolean logic operators *and*, *or*, and *not* into complex conditional expressions. All matching operations return a true or false result. The execution of these conditional expressions and their associated actions can then be controlled by using simple *if then*, *elseif*, and *else* structures, which allow the evaluation paths through the policy to be fully specified by the user.

Routing Policy Language Structure

This section describes the basic structure of RPL.

Names

The policy language provides two kinds of persistent, namable objects: sets and policies. Definition of these objects is bracketed by beginning and ending command lines. For example, to define a policy named test, the configuration syntax would look similar to the following:

```
route-policy test
   [ . . . policy statements . . . ]
end-policy
```

Legal names for policy objects can be any sequence of the upper- and lowercase alphabetic characters; the numerals 0 to 9; and the punctuation characters period, hyphen, and underscore. A name must begin with a letter or numeral.

Sets

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are not allowed.

Four kinds of sets exist: as-path-set, community-set, extcommunity-set, and prefix-set. You may want to perform comparisons against a small number of elements, such as two or three community values, for example. To allow for these comparisons, the user can enumerate these values directly. These enumerations are referred to as *inline sets*. Functionally, inline sets are equivalent to named sets, but allow for simple tests to be inline. Thus, comparisons do not require that a separate named set be maintained when only one or two elements are being compared. See the set types described in the following sections for the syntax. In general, the syntax for an inline set is a comma-separated list surrounded by parentheses as follows: (<element-entry>,<element-entry>,<element-entry>, ...<element-entry>), where <element-entry> is an entry of an item appropriate to the type of usage such as a prefix or a community value.

The following is an example using an inline community set:

```
route-policy sample-inline
    if community matches-any ([10..15]:100) then
        set local-preference 100
    endif
end-policy
```

The following is an equivalent example using the named set test-communities:

```
community-set test-communities
    10:100,
    11:100,
    12:100,
    13:100,
    14:100,
    15:100
end-set

route-policy sample
    if community matches-any test-communities then
        set local-preference 100
    endif
end-policy
```

Both of these policies are functionally equivalent, but the inline form does not require the configuration of the community set just to store the six values. You can choose the form appropriate to the configuration context. In the following sections, examples of both the named set version and the inline form are provided where appropriate.

as-path-set

An AS path set comprises operations for matching an AS path attribute. The only matching operation is a regular expression match.

Named Set Form

The named set form uses the **ios-regex** keyword to indicate the type of regular expression and requires single quotation marks around the regular expression.

The following is a sample definition of a named AS path set:

```
as-path-set aset1
    ios-regex ’_42$’,
    ios-regex ’_127$’
end-set
```

This AS path set comprises two elements. When used in a matching operation, this AS path set matches any route whose AS path ends with either the autonomous system (AS) number 42 or 127.

To remove the named AS path set, use the **no as-path-set aset1** command-line interface (CLI) command.

Inline Set Form

The inline set form is a parenthesized list of comma-separated expressions, as follows:

```
(ios-regex '_42$', ios-regex '_127$')
```

This set matches the same AS paths as the previously named set, but does not require the extra effort of creating a named set separate from the policy that uses it.

community-set

A community-set holds community values for matching against the BGP community attribute. A community is a 32-bit quantity. Integer community values *must* be split in half and expressed as two unsigned decimal integers in the range from 0 to 65535, separated by a colon. Single 32-bit community values are not allowed. The following is the named set form:

Named Set Form

```
community-set cset1
    12:34,
    12:56,
    12:78,
    internet
end-set
```

Inline Set Form

```
(12:34, 12:56, 12:78)
($as:34, $as:$tag1, 12:78, internet)
```

The inline form of a community-set also supports parameterization. Each 16-bit portion of the community may be parameterized. See the "Parameterization" section on page RC-214 for more information.

RPL provides symbolic names for the standard well-known community values: internet is 0:0, no-export is 65535:65281, no-advertise is 65535:65282, and local-as is 65535:65283.

RPL also provides a facility for using *wildcards* in community specifications. A wildcard is specified by inserting an asterisk (*) in place of one of the 16-bit portions of the community specification; the wildcard indicates that any value for that portion of the community matches. Thus, the following policy matches all communities in which the autonomous system part of the community is 123:

```
community-set cset3
   123:*
end-set
```

Every community set must contain at least one community value. Empty community sets are invalid and are rejected.

extcommunity-set

An extended community-set is analogous to a community-set except that it contains extended community values instead of regular community values. It also supports named forms and inline forms. The following are syntactic examples:

Named Form

```
extcommunity-set extcomm-set1
    RT:1.2.3.4:666,
    RT:1234:666,
    SoO:1.2.3.4:777,
    SoO :4567:777
end-set
```

Inline Form

```
(RT:1.2.3.4:666, RT:1234:6667, SoO:1.2.3.4:777, SoO:45678:777)
(RT:$ipaddr:666, RT:1234:$tag, SoO:1.2.3.4:777, SoO:$tag2:777)
```

As with community sets, the inline form supports parameterization within parameterized policies. Either portion of the extended community value can be parameterized.

Every extended community-set must contain at least one extended community value. Empty extended community-sets are invalid and rejected.

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The address is a standard dotted-decimal IPv4 or colon-separated hexadecimal IPv6 address. The mask length, if present, is a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6) following the address and separated from it by a slash. The optional minimum matching length follows the address and optional mask length and is expressed as the keyword **ge** (mnemonic for **g**reater than or **e**qual to), followed by a nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). The optional maximum matching length follows the rest and is expressed by the keyword **le** (mnemonic for **l**ess than or **e**qual to), followed by yet another nonnegative decimal integer in the range from 0 to 32 (0 to 128 for IPv6). A syntactic shortcut for specifying an exact length for prefixes to match is the **eq** keyword (mnemonic for **eq**ual to).

If a prefix match specification has no mask length, then the default mask length is 32 for IPv4 and 128 for IPv6. The default minimum matching length is the mask length. If a minimum matching length is specified, then the default maximum matching length is 32 for IPv4 and 128 for IPv6. Otherwise, if neither minimum nor maximum is specified, the default maximum is the mask length.

The prefix-set itself is a comma-separated list of prefix match specifications. The following are examples:

```
prefix-set legal-ipv4-prefix-examples
    10.0.1.1,
    10.0.2.0/24,
    10.0.3.0/24 ge 28,
    10.0.4.0/24 le 28,
    10.0.5.0/24 ge 26 le 30,
    10.0.6.0/24 eq 28
end-set

prefix-set legal-ipv6-prefix-examples
  2001:0:0:1::/64,
  2001:0:0:2::/64 ge 96,
  2001:0:0:2::/64 ge 96 le 100,
  2001:0:0:2::/64 eq 100
end-set
```

The first element of the prefix-set matches only one possible value, 10.0.1.1/32 or the host address 10.0.1.1. The second element matches only one possible value, 10.0.2.0/24. The third element matches a range of prefix values, from 10.0.3.0/28 to 10.0.3.255/32. The fourth element matches a range of values, from 10.0.4.0/24 to 10.0.4.240/28. The fifth element matches prefixes in the range from 10.0.5.0/26 to 10.0.5.252/30. The sixth element matches any prefix of length 28 in the range from 10.0.6.0/28 through 10.0.6.240/28.

The following prefix-set consists entirely of invalid prefix match specifications:

```
prefix-set ILLEGAL-PREFIX-EXAMPLES
    10.1.1.1 ge 16,
    10.1.2.1 le 16,
    10.1.3.0/24 le 23,
    10.1.4.0/24 ge 33,
    10.1.5.0/25 ge 29 le 28
end-set
```

Neither the minimum length nor maximum length is valid without a mask length. The maximum length must be at least the mask length. For IPv4, the minimum length must be less than 32, the maximum length of an IPv4 prefix. For IPv6, the minimum length must be less than 128, the maximum length of an IPv6 prefix. The maximum length must be equal to or greater than the minimum length.

Routing Policy Language Components

Four main components in the routing policy language are involved in defining, modifying, and using policies: the configuration front end, policy repository, execution engine, and policy clients themselves.

The configuration front end (CLI) is the mechanism to define and modify policies. This configuration is then stored on the router using the normal storage means and can be displayed using the normal configuration **show** commands.

The second component of the policy infrastructure, the policy repository, has several responsibilities. First, it compiles the user-entered configuration into a form that the execution engine can understand. Second, it performs much of the verification of policies; and it ensures that defined policies can actually be executed properly. Third, it tracks which attach points are using which policies so that when policies are modified the appropriate clients are properly updated with the new policies relevant to them.

The third component is the execution engine. This component is the piece that actually runs policies as the clients request. The process can be thought of as receiving a route from one of the policy clients and then executing the actual policy against the specific route data.

The fourth component is the policy clients (the routing protocols). This component calls the execution engine at the appropriate times to have a given policy be applied to a given route, and then perform some number of actions. These actions may include deleting the route if policy indicated that it should be dropped, passing along the route to the protocol decision tree as a candidate for the best route, or advertising a policy modified route to a neighbor or peer as appropriate.

Routing Policy Language Usage

This section provides basic routing policy language usage examples. See the "How to Implement Routing Policy" section on page RC-237 for detailed information on how to implement routing policy language.

The *pass* policy

The following example shows how the policy accepts all presented routes without modifying the routes.

```
route-policy quickstart-pass
    pass
end-policy
```

The *drop everything* policy

The following example shows how the policy explicitly rejects all routes presented to it. This type of policy is used to ignoring everything coming from a misbehaving peer.

```
route-policy quickstart-drop
    drop
end-policy
```

Ignore routes with specific AS numbers in the path

The following example shows the policy definition in three parts. First, the **as-path-set** command defines three regular expressions to match against an AS path. Second, the **route-policy** command applies the AS path set to a route. If the AS path attribute of the route matches the regular expression defined with the **as-path-set** command, the protocol refuses the route. Third, the route policy is attached to BGP neighbor 10.0.1.2. BGP consults the policy named ignore_path_as on routes received (imported) from neighbor 10.0.1.2.

```
as-path-set ignore_path
    ios-regex '_11_',
    ios-regex '_22_',
    ios-regex '_33_'
end-set

route-policy ignore_path_as
    if as-path in ignore_path then
        drop
    else
        pass
    endif
end-policy

router bgp 2
    neighbor 10.0.1.2 address-family ipv4 unicast policy ignore_path_as in
```

Set community based on MED

The following example shows how the policy tests the MED of a route and modifies the community attribute of the route based on the value of the MED. If the MED value is 127, the policy adds the community 123:456 to the route. If the MED value is 63, the policy adds the value 123:789 to the community attribute of the route. Otherwise, the policy removes the community 123:123 from the route. In any case, the policy instructs the protocol to accept the route.

```
route-policy quickstart-med
    if med eq 127 then
        set community (123:456) additive
    elseif med eq 63 then
        set community (123:789) additive
    else
        delete community in (123:123)
    endif
    pass
end-policy
```

Set local preference based on community

The following example shows how the community-set named quickstart-communities defines community values. The route policy named quickstart-localpref tests a route for the presence of the communities specified in the quickstart-communities community set. If any of the community values are present in the route, the route policy sets the local preference attribute of the route to 31. In any case, the policy instructs the protocol to accept the route.

```
community-set quickstart-communities
    987:654,
    987:543,
    987:321,
    987:210
end-set
```

```
route-policy quickstart-localpref
    if community matches-any quickstart-communities then
        set local-preference 31
    endif
    pass
end-policy
```

Persistent Remarks

The following example shows how comments are placed in the policy to clarify the meaning of the entries in the set and the statements in the policy. The remarks are persistent, meaning they remain attached to the policy. For example, remarks are displayed in the output of the **show running-config** command. Adding remarks to the policy makes the policy easier to understand, modify at a later date, and troubleshoot if an unexpected behavior occurs.

```
prefix-set rfc1918
    # These are the networks defined as private in RFC1918 (including
    # all subnets thereof)
    10.0.0.0/8 ge 8,
    172.16.0.0/12 ge 12,
    192.168.0.0/16 ge 16
end-set

route-policy quickstart-remarks
    # Handle routes to RFC1918 networks
    if destination in rfc1918 then
        # Set the community such that we do not export the route
        set community (no-export) additive
    endif
end-policy
```

Routing Policy Configuration Basics

Route policies comprise series of statements and expressions that are bracketed with the **route-policy** and **end-policy** keywords. Rather than a collection of individual commands (one for each line), the statements within a route policy have context relative to each other. Thus, instead of each line being an individual command, each policy or set is an independent configuration object that can be used, entered, and manipulated as a unit.

Each line of a policy configuration is a logical subunit. At least one new line must follow the **then**, **else**, and **end-policy** keywords. A new line must also follow the closing parenthesis of a parameter list and the name string in a reference to an AS path set, community set, extended community set, or prefix set. At least one new line must precede the definition of a route policy, AS path set, community set, extended community set, or prefix set. One or more new lines can follow an action statement. One or more new lines can follow a comma separator in a named AS path set, community set, extended community set, or prefix set. A new line must appear at the end of a logical unit of policy expression and may not appear anywhere else.

Policy Definitions

Policy definitions create named sequences of policy statements. A policy definition consists of the CLI **route-policy** keyword followed by a name, a sequence of policy statements, and the **end-policy** keyword. For example, the following policy drops any route it encounters:

```
route-policy drop-everything
    drop
end-policy
```

The name serves as a handle for binding the policy to protocols. To remove a policy definition, issue the **no route-policy** *name* command.

Policies may also refer to other policies such that common blocks of policy can be reused. This reference to other policies is accomplished by using the **apply** statement, as shown in the following example:

```
route-policy check-as-1234
    if as-path passes-through ‘1234’ then
        apply drop-everything
    else
        pass
    endif
end-policy
```

The **apply** statement indicates that the policy drop-everything should be executed if the route under consideration passed through autonomous system 1234 before it is received. If a route that has autonomous system 1234 in its AS path is received, the route is dropped; otherwise, the route is accepted without modification. This policy is an example of a hierarchical policy. Thus, the semantics of the **apply** statement are just as if the applied policy were cut and pasted into the applying policy:

```
route-policy check-as-1234-prime
 if as-path passes-through '1234' then
    drop
 else
    pass
 endif
end-policy
```

You may have as many levels of hierarchy as desired. However, many levels may be difficult to maintain and understand.

Parameterization

In addition to supporting reuse of policies using the **apply** statement, policies can be defined that allow for parameterization of some of the attributes. The following example shows how to define a parameterized policy named param-example. In this case, the policy takes one parameter, $mytag. Parameters always begin with a dollar sign and consist otherwise of any alphanumeric characters. Parameters can be substituted into any attribute that takes a parameter.

In the following example, a 16-bit community tag is used as a parameter:

```
route-policy param-example ($mytag)
        set community (1234:$mytag) additive
end-policy
```

This parameterized policy can then be reused with different parameterizations, as shown in the following example. In this manner, policies that share a common structure but use different values in some of their individual statements can be modularized. For details on which attributes can be parameterized, see the individual attribute sections.

```
route-policy origin-10
    if as-path originates-from '10' then
        apply param-example(10)
    else
        pass
    endif
end-policy

route-policy origin-20
    if as-path originates-from '20' then
        apply param-example(20)
    else
        pass
    endif
end-policy
```

The parameterized policy param-example provides a policy definition that is expanded with the values provided as the parameters in the apply statement. Note that the policy hierarchy is always maintained, Thus, if the definition of param-example changes, then the behavior of origin_10 and origin_20 changes to match.

The effect of the origin-10 policy is that it adds the community 1234:10 to all routes that pass through this policy and have an AS path indicating the route originated from autonomous system 10. The origin-20 policy is similar except that it adds to community 1234:20 for routes originating from autonomous system 20.

Semantics of Policy Application

This section discusses how routing policies are evaluated and applied. The following concepts are discussed:

- Boolean Operator Precedence, page RC-215
- Multiple Modifications of the Same Attribute, page RC-216
- When Attributes Are Modified, page RC-216
- Default Drop Disposition, page RC-217
- Control Flow, page RC-217
- Policy Verification, page RC-218

Boolean Operator Precedence

Boolean expressions are evaluated in order of operator precedence, from left to right. The highest precedence operator is *not*, followed by *and*, and then *or*. The following expression:

```
med eq 10 and not destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

if fully parenthesized to display the order of evaluation, would look like this:

```
(med eq 10 and (not destination in (10.1.3.0/24))) or community matches-any ([10..25]:35)
```

The inner *not* applies only to the destination test; the *and* combines the result of the *not* expression with the Multi Exit Discriminator (MED) test; and the *or* combines that result with the community test. If the order of operations are rearranged:

```
not med eq 10 and destination in (10.1.3.0/24) or community matches-any ([10..25]:35)
```

then the expression, fully parenthesized, would look like the following:

```
((not med eq 10) and destination in (10.1.3.0/24)) or community matches-any ([10..25]:35)
```

Multiple Modifications of the Same Attribute

When a policy replaces the value of an attribute multiple times, the last assignment wins because all actions are executed. Because the MED attribute in BGP is one unique value, the last value to which it gets set to wins. Therefore, the following policy results in a route with a MED value of 12:

```
set med 9
set med 10
set med 11
set med 12
```

This example is trivial, but the feature is not. It is possible to write a policy that effectively changes the value for an attribute. For example:

```
set med 8
if community matches-any cs1 then
    set local-preference 122
    if community matches-any cs2 then
        set med 12
    endif
endif
```

The result is a route with a MED of 8, unless the community list of the route matches both cs1 and cs2, in which case the result is a route with a MED of 12.

In the case in which the attribute being modified can contain only one value, it is easy to think of this case as the last statement wins. However, a few attributes can contain multiple values and the result of multiple actions on the attribute is cumulative rather than as a replacement. The first of these cases is the use of the **additive** keyword on community and extended community evaluation. Consider a policy of the form:

```
route-policy community-add
    set community (10:23)
    set community (10:24) additive
    set community (10:25) additive
end-policy
```

This policy sets the community string on the route to contain all three community values: 10:23, 10:24, and 10:25.

The second of these cases is AS path prepending. Consider a policy of the form:

```
route-policy prepend-example
    prepend as-path 2 3
    prepend as-path 666 2
end-policy
```

This policy prepends the following to the AS path (666 666 2 2 2). This prepending is a result of all actions being taken and to AS path being an attribute that contains an array of values rather than a simple scalar value.

When Attributes Are Modified

A policy does not modify route attribute values until all tests have been completed. In other words, comparison operators always run on the initial data in the route. Intermediate modifications of the route attributes do not have a cascading effect on the evaluation of the policy. Take the following example:

```
if  med eq 12 then
    set med 42
    if med eq 42 then
        drop
    endif
endif
```

This policy never executes the drop statement because the second test (med eq 42) sees the original, unmodified value of the MED in the route. Because the MED has to be 12 to get to the second test, the second test always returns false.

Default Drop Disposition

All route policies have a default action to drop the route under evaluation unless the route has been modified by a policy action or explicitly passed. Applied (nested) policies implement this disposition as though the applied policy were pasted into the point where it is applied.

Consider a policy to allow all routes in the 10 network and set their local preference to 200 while dropping all other routes. You might write the policy as follows:

```
route-policy two
    if destination in (10.0.0.0/8 ge 8 le 32) then
        set local-preference 200
    endif
end-policy

route-policy one
    apply two
end-policy
```

It may appear that policy one drops all routes because it neither contains an explicit **pass** statement nor modifies a route attribute. However, the applied policy does set an attribute for some routes and this disposition is passed along to policy one. The result is that policy one passes routes with destinations in network 10, and drops all others.

Control Flow

Policy statements are processed sequentially in the order in which they appear in the configuration. Policies that hierarchically reference other policy blocks are processed as if the referenced policy blocks had been directly substituted inline. For example, if the following policies are defined:

```
route-policy one
    set weight 100
end-policy

route-policy two
    set med 200
end-policy

route-policy three
    apply two
    set community (2:666) additive
end-policy

route-policy four
    apply one
    apply three
    pass
end-policy
```

Policy four could be rewritten in an equivalent way as follows:

```
route-policy four-equivalent
    set weight 100
    set med 200
    set community (2:666) additive
    pass
end-policy
```

Note The **pass** statement is not required and can be removed to represent the equivalent policy in another way.

Policy Verification

Several different types of verification occur when policies are being defined and used.

Range Checking

As policies are being defined, some simple verifications, such as range checking of values, is done. For example, the MED that is being set is checked to verify that it is in a proper range for the MED attribute. However, this range checking cannot cover parameter specifications because they may not have defined values yet. These parameter specifications are verified when a policy is attached to an attach point. The policy repository also verifies that there are no recursive definitions of policy, and that parameter numbers are correct. At attach time, all policies must be well formed. All sets and policies that they reference must be defined and have valid values. Likewise, any parameter values must also be in the proper ranges.

Incomplete Policy and Set References

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies, which allows for freedom of workflow. You can build configurations that reference sets or policy blocks that are not yet defined, and then can later fill in those undefined policies and sets, thereby achieving much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, a user can define a policy sample that references the policy bar using an **apply** statement even if the policy bar does not exist. Similarly, a user can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. If you attempt to attach the policy sample with the reference to an undefined policy bar at an inbound BGP policy using the **neighbor 1.2.3.4 address-family ipv4 unicast policy sample in** command, the configuration attempt is rejected because the policy bar does not exist.

Likewise, you cannot remove a route policy or set that is currently in use at an attach point because this removal would result in an undefined reference. An attempt to remove a route policy or set that is currently in use results in an error message to the user.

A condition exists that is referred to as a null policy in which the policy bar exists but has no statements, actions, or dispositions in it. In other words, the policy bar does exist as follows:

```
route-policy bar
end-policy
```

This is a valid policy block. It effectively forces all routes to be dropped because it is a policy block that never modifies a route, nor does it include the pass statement. Thus, the default action of drop for the policy block is followed.

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. Traditionally, configuration changes are done by completely removing the relevant configuration and then re-entering it. However, this allows for a window of time in which no policy is attached and the default action takes place. RPL provides a mechanism for an atomic change so that if a policy is redeclared, or edited using the emacs editor, the new configuration is applied immediately, which allows for policies that are in use to be changed without having a window of time in which no policy is applied at the given attach point.

Verification of Attribute Comparisons and Actions

The policy repository knows which attributes, actions, and comparisons are valid at each attach point. When a policy is attached, these actions and comparisons are verified against the capabilities of that particular attach point. Take, for example, the following policy definition:

```
route-policy bad
    set med 100
    set level level-1-2
    set cost 200
end-policy
```

This policy attempts to perform actions to set the BGP attribute med, IS-IS attribute level, and OSPF attribute cost. The system allows you to define such a policy, but it does not allow you to attach such a policy. If you had defined the policy bad and then attempted to attach it as an inbound BGP policy using the BGP configuration statement **neighbor 1.2.3.4 address-family ipv4 unicast route-policy bad in** the system would reject this configuration attempt. This rejection results from the verification process checking the policy and realizing that while BGP could set the MED, it has no way of setting the level or cost as the level and cost are attributes of IS-IS and OSPF, respectively. Instead of silently omitting the actions that cannot be done, the system generates an error to the user. Likewise, a valid policy in use at an attach point cannot be modified in such a way as to introduce an attempt to modify a nonexistent attribute or to compare against a nonexistent attribute. The verifiers test for nonexistent attributes and reject such a configuration attempt.

Policy Statements

Four types of policy statements exist: remark, disposition (drop and pass), action (set), and if (comparator).

Remark

A remark is text attached to policy configuration but otherwise ignored by the policy language parser. Remarks are useful for documenting parts of a policy. The syntax for a remark is text that has each line prepended with a pound sign (#):

```
# This is a simple one-line remark.

# This
# is a remark
# comprising multiple
# lines.
```

In general, remarks are used between complete statements or elements of a set. Remarks are not supported in the middle of statements or within an inline set definition.

Unlike traditional !-comments in the CLI, RPL remarks persist through reboots and when configurations are saved to disk or a TFTP server and then loaded back onto the router.

Disposition

By default, a route is **dropped** at the end of policy processing unless either the policy **modifies** a route attribute or it passes the route by means of an explicit **pass** statement. For example, the following policy drops all routes because it neither modifies the attribute of any route nor explicitly passes it.

```
route-policy EMPTY
end-policy
```

Whereas the following policies pass all routes that they evaluate.

```
route-policy PASS-ALL
    pass
end-policy


route-policy SET-LPREF
    set local-preference 200
end-policy
```

In addition to being implicitly dropped, a route may be dropped by an **explicit drop** statement. **Drop** statements cause a route to be dropped immediately so that no further policy processing is done. Note also that a **drop** statement overrides any previously processed **pass** statements or attribute modifications. For example, the following policy drops all routes. The first **pass** statement is executed, but is then immediately overridden by the **drop** statement. The second **pass** statement never gets executed.

```
route-policy DROP-EXAMPLE
    pass
    drop
    pass
end-policy
```

When one policy applies another, it is as if the applied policy were copied into the right place in the applying policy, and then the same drop-and-pass semantics are put into effect. For example, policies ONE and TWO are equivalent to policy ONE-PRIME:

```
route-policy ONE
    apply route-policy two
    if as-path neighbor-is '123' then
        pass
    endif
end-policy

route-policy TWO
    if destination in (10.0.0.0/16 le 32) then
        drop
    endif
end-policy

route-policy ONE-PRIME
    if destination in (10.0.0.0/16 le 32) then
        drop
    endif
    if as-path neighbor-is '123' then
        pass
    endif
end-policy
```

Because the effect of an **explicit drop** statement is immediate, routes in 10.0.0.0/16 le 32 are dropped without any further policy processing. Other routes are then considered to see if they were advertised by autonomous system 123. If they were advertised, they are passed; otherwise, they are implicitly dropped at the end of all policy processing.

Action

An action is a sequence of primitive operations that modify a route. Most actions, but not all, are distinguished by the **set** keyword. In a route policy, actions can be grouped together. For example, the following is a route policy comprising three actions:

```
route-policy actions
    set med 217
    set community (12:34) additive
    delete community in (12:56)
end-policy
```

If

In its simplest form, an **if** statement uses a conditional expression to decide which actions or dispositions should be taken for the given route. For example:

```
if as-path in as-path-set-1 then
    drop
endif
```

The example indicates that any routes whose AS path is in the set as-path-set-1 are dropped. The contents of the **then** clause may be an arbitrary sequence of policy statements.

The following example contains two action statements:

```
if origin is igp then
    set med 42
    prepend as-path 73 5
endif
```

The CLI provides support for the **exit** command as an alternative to the **endif** command.

The **if** statement also permits an **else** clause, which is executed if the if condition is false:

```
if med eq 8 then
    set community (12:34) additive
else
    set community (12:56) additive
endif
```

The policy language also provides syntax, using the **elseif** keyword, to string together a sequence of tests:

```
if med eq 150 then
    set local-preference 10
elseif med eq 200 then
    set local-preference 60
elseif med eq 250 then
    set local-preference 110
else
    set local-preference 0
endif
```

The statements within an **if** statement may themselves be **if** statements, as shown in the following example:

```
if community matches-any (12:34,56:78) then
    if med eq 150 then
        drop
    endif
    set local-preference 100
endif
```

This policy example sets the value of the local preference attribute to 100 on any route that has a community value of 12:34 or 56:78 associated with it. However, if any of these routes has a MED value of 150, then these routes with either the community value of 12:34 or 56:78 and a MED of 150 are dropped.

Boolean Conditions

In the previous section describing the **if** statement, all of the examples use simple Boolean conditions that evaluate to either true or false. RPL also provides a way to build compound conditions from simple conditions by means of Boolean operators.

Three Boolean operators exist: negation (**not**), conjunction (**and**), and disjunction (**or**). In the policy language, negation has the highest precedence, followed by conjunction, and then by disjunction. Parentheses may be used to group compound conditions to override precedence or to improve readability.

The following simple condition:

```
med eq 42
```

is true only if the value of the MED in the route is 42, otherwise it is false.

A simple condition may also be negated using the **not** operator:

```
not next-hop in (10.0.2.2)
```

Any Boolean condition enclosed in parentheses is itself a Boolean condition:

```
(destination in prefix-list-1)
```

A compound condition takes either of two forms. It can be a simple expression followed by the **and** operator, itself followed by a simple condition:

```
med eq 42 and next-hop in (10.0.2.2)
```

A compound condition may also be a simpler expression followed by the **or** operator and then another simple condition:

```
origin is igp or origin is incomplete
```

An entire compound condition may be enclosed in parentheses:

```
(med eq 42 and next-hop in (10.0.2.2))
```

The parentheses may serve to make the grouping of subconditions more readable, or they may force the evaluation of a subcondition as a unit.

In the following example, the highest-precedence **not** operator applies only to the destination test, the **and** operator combines the result of the **not** expression with the community test, and the **or** operator combines that result with the MED test.

```
med eq 10 or not destination in (10.1.3.0/24) and community matches-any
([12..34]:[56..78])
```

With a set of parentheses to express the precedence, the result is the following:

```
med eq 10 or ((not destination in (10.1.3.0/24)) and community matches-any
([12..34]:[56..78])
```

The following is another example of a complex expression:

```
(origin is igp or origin is incomplete or not med eq 42) and next-hop in (10.0.2.2)
```

The left conjunction is a compound condition enclosed in parentheses. The first simple condition of the inner compound condition tests the value of the origin attribute; if it is Interior Gateway Protocol (IGP), then the inner compound condition is true. Otherwise, the evaluation moves on to test the value of the origin attribute again, and if it is incomplete, then the inner compound condition is true. Otherwise, the evaluation moves to check the next component condition, which is a negation of a simple condition.

apply

As discussed in the sections on policy definitions and parameterization of policies, the **apply** command executes another policy (either parameterized or unparameterized) from within another policy, which allows for the reuse of common blocks of policy. When combined with the ability to parameterize common blocks of policy, the **apply** command becomes a powerful tool for reducing repetitive configuration.

Attach Points

Policies do not become useful until they are applied to routes, and for policies to be applied to routes they need to be made known to routing protocols. In BGP, for example, there are several situations where policies can be used, the most common of these is defining import and export policy. The policy attach point is the point in which an association is formed between a specific protocol entity, in this case a BGP neighbor, and a specific named policy. It is important to note that a verification step happens at this point. Each time a policy is attached, the given policy and any policies it may apply are checked to ensure that the policy can be validly used at that attach point. For example, if a user defines a policy that sets the IS-IS level attribute and then attempts to attach this policy as an inbound BGP policy, the attempt would be rejected because BGP routes do not carry IS-IS attributes. Likewise, when policies are modified that are in use, the attempt to modify the policy is verified against all current uses of the policy to ensure that the modification is compatible with the current uses.

Each protocol has a distinct definition of the set of attributes (commands) that compose a route. For example, BGP routes may have a community attribute, which is undefined in OSPF. Routes in IS-IS have a level attribute, which is unknown to BGP. Routes carried internally in the RIB may have a tag attribute.

When a policy is attached to a protocol, the protocol checks the policy to ensure the policy operates using route attributes known to the protocol. If the protocol uses unknown attributes, then the protocol rejects the attachment. For example, OSPF rejects attachment of a policy that tests the values of BGP communities.

The situation is made more complex by the fact that each protocol has access to at least two distinct route types. In addition to native protocol routes, for example BGP or IS-IS, some protocol policy attach points operate on RIB routes, which is the common central representation. Using BGP as an example, the protocol provides an attach point to apply policy to routes redistributed from the RIB to BGP. An attach point dealing with two different kinds of routes permits a mix of operations: RIB attribute operations for matching and BGP attribute operations for setting.

Note The protocol configuration rejects attempts to attach policies that perform unsupported operations.

The following sections describe the protocol attach points, including information on the attributes (commands) and operations that are valid for each attach point.

- BGP Policy Attach Points, page RC-224
- OSPF Policy Attach Points, page RC-231
- OSPFv3 Policy Attach Points, page RC-232
- IS-IS Policy Attach Points, page RC-234

See the *Cisco IOS XR Routing Command Reference* for more information on the attributes and operations.

BGP Policy Attach Points

This section describes each of the BGP policy attach points and provides a summary of the BGP attributes and operators.

Aggregation

The aggregation attach point generates an aggregate route to be advertised based on the conditional presence of subcomponents of that aggregate. Policies attached at this attach point are also able to set any of the valid BGP attributes on the aggregated routes. For example, the policy could set a community value or a MED on the aggregate that is generated. The specified aggregate is generated if any routes evaluated by the named policy pass the policy. More specifics of the aggregate are filtered using the **suppress-route** keyword. Any actions taken to set attributes in the route affect attributes on the aggregate.

In the policy language, the configuration is controlled by which routes pass the policy. The suppress map was used to selectively filter or suppress specific components of the aggregate when the summary-only flag is not set. In other words, when the aggregate and more specific components are being sent, some of the more specific components can be filtered using a suppress map. In the policy language, this is controlled by selecting the route and setting the suppress flag. The attribute-map allowed the user to set specific attributes on the aggregated route. In the policy language, setting attributes on the aggregated route is controlled by normal action operations.

In the following example, the aggregate address 10.0.0.0/8 is generated if there are any component routes in the range 10.0.0.0/8 ge 8 le 25 except for 10.2.0.0/24. Because summary-only is not set, all components of the aggregate are advertised. However, the specific component 10.1.0.0 are suppressed.

```
route-policy sample
    if destination in (10.0.0.0/8 ge 8 le 25) then
        set community (10:33)
    endif
    if destination in (10.2.0.0/24) then
        drop
    endif
    if destination in (10.1.0.0/24) then
         suppress-route
    endif
end-policy
```

```
router bgp 2
address-family ipv4
    aggregate-address 10.0.0.0/8 policy sample
    .
    .
    .
```

Dampening

The dampening attach point controls the default route-dampening behavior within BGP. Unless overridden by a more specific policy on the associate peer, all routes in BGP apply the associated policy to set their dampening attributes.

The following policy sets dampening values for BGP IPv4 unicast routes. Those routes that are more specific than a /25 take longer to recover after they have been dampened than routes that are less specific than /25.

```
route-policy sample_damp
    if destination in (0.0.0.0/0 ge 25) then
        set dampening halflife 30 others default
    else
        set dampening halflife 20 others default
    endif
end-policy

router bgp 2
    address-family ipv4 unicast
        bgp dampening policy sample_damp
        .
        .
        .
```

Default Originate

The default originate attach point allows the default route (0.0.0.0/0) to be conditionally generated and advertised to a peer, based on the presence of other routes. It accomplishes this configuration by evaluating the associated policy against routes in the Routing Information Base (RIB). If any routes pass the policy, the default route is generated and sent to the relevant peer.

The following policy generates and sends a default-route to the BGP neighbor 10.0.0.1 if any routes that match 10.0.0.0/8 ge 8 le 32 are present in the RIB.

```
route-policy sample-originate
    if rib-has-route in (10.0.0.0/8 ge 8 le 32) then
        pass
    endif
end-policy

router bgp 2
    neighbor 10.0.0.1
            remote-as 3
            address-family ipv4 unicast
            default-originate policy sample-originate
            .
            .
            .
```

Note The current implementation of default origination policy permits matching only on destination address.

Neighbor Export

The neighbor export attach point selects the BGP routes to send to a given peer or group of peers. The routes are selected by running the set of possible BGP routes through the associated policy. Any routes that pass the policy are then sent as updates to the peer or group of peers. The routes that are sent may have had their BGP attributes altered by the policy that has been applied.

The following policy sends all BGP routes to neighbor 10.0.0.5. Routes that are tagged with any community in the range 2:100 to 2:200 are sent with a MED of 100 and a community of 2:666. The rest of the routes are sent with a MED of 200 and a community of 2:200.

```
route-policy sample-export
    if community matches-any (2:[100-200]) then
        set med 100
        set community (2:666)
    else
        set med 200
        set community (2:200)
    endif
end-policy

router bgp 2
    neighbor 10.0.0.5
    remote-as 3
    address-family ipv4 unicast
        route-policy sample-export out
        .
        .
        .
```

Neighbor Import

The neighbor import attach point controls the reception of routes from a specific peer. All routes that are received by a peer are run through the attached policy. Any routes that pass the attached policy are passed to the BGP Routing Information Base (BRIB) as possible candidates for selection as best path routes.

When a BGP import policy is modified, it is necessary to rerun all the routes that have been received from that peer against the new policy. The modified policy may now discard routes that were previously allowed through, allow through previously discarded routes, or change the way the routes are modified. A new configuration option in BGP (**bgp auto-policy-soft-reset**) that allows this modification to happen automatically in cases for which either soft reconfiguration is configured or the BGP route-refresh capability has been negotiated.

The following example shows how to receive routes from neighbor 10.0.0.1. Any routes received with the community 3:100 have their local preference set to 100 and their community tag set to 2:666. All other routes received from this peer have their local preference set to 200 and their community tag set to 2:200.

```
route-policy sample_import
    if community matches-any (3:100) then
        set local-preference 100
        set community (2:666)
    else
        set local-preference 200
        set community (2:200)
    endif
end-policy
```

```
router bgp 2
    neighbor 10.0.0.1
        remote-as 3
        address-family ipv4 unicast
            route-policy sample_import in
            .
            .
            .
```

Network

The network attach point controls the injection of routes from the RIB into BGP. A route policy attached at this point is able to set any of the valid BGP attributes on the routes that are being injected.

The following example shows a route policy attached at the network attach point that sets the well-known community no-export for any routes more specific than /24:

```
route-policy NetworkControl
   if destination in (0.0.0.0/0 ge 25) then
      set community (no-export) additive
   endif
end-policy

router bgp 2
    address-family ipv4 unicast
    network 172.16.0.5/27 route-policy NetworkControl
```

Redistribute

The redistribute attach point allows routes from other sources to be advertised by BGP. The policy attached at this point is able to set any of the valid BGP attributes on the routes that are being redistributed. Likewise, selection operators allow a user to control what route sources are being redistributed and which routes from those sources.

The following example shows how to redistribute all routes from OSPF instance 12 into BGP. If OSPF were carrying a default route, it is dropped. Routes carrying a tag of 10 have their local preference set to 300 and the community value of 2:666 and no-advertise attached. All other routes have their local preference set to 200 and a community value of 2:100 set.

```
route-policy sample_redistribute
    if destination in (0.0.0.0/0) then
        drop
    endif
    if tag eq 10 then
        set local-preference 300
        set community (2:666, no-advertise)
    else
        set local-preference 200
        set community (2:100)
    endif
end-policy

router bgp 2
    address-family ipv4 unicast
        redistribute ospf 12 route-policy sample_redistribute
        .
        .
```

Show bgp

The show bgp attach point allows the user to display selected BGP routes that pass the given policy. Any routes that are not dropped by the attached policy are displayed in a manner similar to the output of the **show ip bgp** command.

In the following example, the **show bgp route-policy** command is used to display any BGP routes carrying a MED of 5:

```
route-policy sample-display
    if med eq 5 then
        pass
    endif
end-policy
!
show bgp route-policy sample-display
```

A **show bgp policy route-policy** command also exists, which runs all routes in the RIB past the named policy as if the RIB were an outbound BGP policy. This command then displays what each route looked like before it was modified and after it was modified, as shown in the following example:

```
RP/0/RP0/CPU0:router# show rpl route-policy test2

route-policy test2
  if (destination in  (10.0.0.0/8 ge 8 le 32)) then
    set med 333
  endif
end-policy
!
RP/0/RP0/CPU0:router# show bgp

BGP router identifier 10.0.0.1, local AS number 2
BGP main routing table version 11
BGP scan interval 60 secs
Status codes:s suppressed, d damped, h history, * valid, > best
              i - internal, S stale
Origin codes:i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.0.0.0           10.0.1.2                10             0 3 ?
*> 10.0.0.0/9         10.0.1.2                10             0 3 ?
*> 10.0.0.0/10        10.0.1.2                10             0 3 ?
*> 10.0.0.0/11        10.0.1.2                10             0 3 ?
*> 10.1.0.0/16        10.0.1.2                10             0 3 ?
*> 10.3.30.0/24       10.0.1.2                10             0 3 ?
*> 10.3.30.128/25     10.0.1.2                10             0 3 ?
*> 10.128.0.0/9       10.0.1.2                10             0 3 ?
*> 10.255.0.0/24      10.0.101.2            1000    555      0 100 e
*> 10.255.64.0/24     10.0.101.2            1000    555      0 100 e
....
```

```
RP/0/RP0/CPU0:router# show bgp policy route-policy test2

10.0.0.0/8 is advertised to 10.0.101.2

  Path info:
    neighbor:10.0.1.2        neighbor router id:10.0.1.2
    valid  external  best
  Attributes after inbound policy was applied:
    next hop:10.0.1.2
    MET ORG AS
    origin:incomplete  neighbor as:3  metric:10
    aspath:3
  Attributes after outbound policy was applied:
    next hop:10.0.1.2
    MET ORG AS
    origin:incomplete  neighbor as:3  metric:333
    aspath:2 3
    ...
```

Table Policy

The table policy attach point allows the user to configure traffic-index values on routes as they are installed into the global routing table. This attach point supports the BGP policy accounting feature. BGP policy accounting uses the traffic indexes that are set on the BGP routes to track various counters. This way, router operators can select different sets of BGP route attributes using the matching operations and then set different traffic indexes for each different class of route they are interested in tracking.

The following example shows how to set the traffic index to 10 in IPv4 unicast routes that originated from autonomous system 10. Likewise, any IPv4 unicast routes that originated from autonomous system 11 have their traffic index set to 11 when they are installed into the FIB. These traffic indexes are then used to count traffic being forwarded on these routes in line cards by enabling the BGP policy accounting counters on the interfaces of interest.

```
route-policy sample-table
    if as-path originates-from ‘10’ then
        set traffic-index 10
    elseif as-path originates-from ‘11’ then
        set traffic-index 11
    endif
end-policy
router bgp 2
    address-family ipv4 unicast
        table-policy sample-table
        .
        .
        .
```

BGP Attributes and Operators

Table 3 summarizes the BGP attributes and operators.

Table 3 BGP Attributes and Operators

Attribute	Match	Set
as-path	in is-local length neighbor-is originates-from passes-though unique-length	prepend
community	is-empty matches-any matches-every	delete set
dampening	n/a	set dampening... to set values that control the dampening (see Dampening, page RC-225)
destination	in	n/a
extended community	is-empty matches-any matches-every	delete set
local-preference	n/a	set
med	is, eq, ge, le	set set + set -
next-hop	in	set
origin	is	set
rib-has-route	in	n/a
route-type	is	n/a
source	in	n/a
suppress-route	n/a	suppress-route
tag	is, eq, ge, le	set
traffic-index	n/a	set
unsuppress-route	n/a	unsuppress-route
weight	n/a	set

Some BGP route attributes are inaccessible from some BGP attach points for various reasons. For example, the **set med igp-cost only** command makes sense when there is a configured igp-cost to provide a source value. Table 4 summarizes which operations are valid and where they are valid.

Table 4 Restricted BGP Operations by Attach Point

	import	**export**	**aggregation**	**redistribution**
prepend as-path	eBGP only	eBGP only	n/a	n/a
set med igp-cost	forbidden	eBGP only	forbidden	forbidden
set weight	n/a	forbidden	n/a	n/a
suppress	forbidden	forbidden	n/a	forbidden

OSPF Policy Attach Points

This section describes each of the OSPF policy attach points and provides a summary of the OSPF attributes and operators.

Default Originate

The default originate attach point allows the user to conditionally inject the default route 0.0.0.0/0 into the OSPF link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 10.0.0.0/8 ge 8 le 25 are present in the RIB:

```
route-policy ospf-originate
   if rib-has-route in (10.0.0.0/8 ge 8 le 25) then
      pass
   endif
end-policy

router ospf 1
   default-information originate policy ospf-originate
   .
   .
   .
```

Redistribute

The redistribute attach point within OSPF injects routes from other routing protocol sources into the OSPF link-state database, which is done by selecting the route types it wants to import from each protocol. It then sets the OSPF parameters of cost and metric type. The policy can control how the routes are injected into OSPF by using the **set level** command.

The following example shows how to redistribute routes from IS-IS instance instance_10 into OSPF instance 1 using the policy OSPF-redist. The policy sets the metric type to type-2 for all redistributed routes. IS-IS routes with a tag of 10 have their cost set to 100, and IS-IS routes with a tag of 20 have their OSPF cost set to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPF link-state database.

```
route-policy OSPF-redist
    set metric-type type-2
    if tag eq 10 then
        set cost 100
    elseif tag eq 20 then
        set cost 200
    else
        drop
    endif
end-policy

router ospf 1
    redistribute isis instance_10 policy OSPF-redist
    .
    .
    .
```

OSPF Attributes and Operators

Table 5 summarizes the OSPF attributes and operators.

Table 5 OSPF Attributes and Operators

Attribute	Match	Set
cost	n/a	set
destination	in	n/a
metric-type	n/a	set
rib-has-route	in	n/a
route-type	is	n/a
tag	eq, ge, le	set

OSPFv3 Policy Attach Points

This section describes each of the OSPFv3 policy attach points and provides a summary of the BGP attributes and operators.

Default Originate

The default originate attach point allows the user to conditionally inject the default route 0::/0 into the OSPFv3 link-state database, which is done by evaluating the attached policy. If any routes in the local RIB pass the policy, then the default route is inserted into the link-state database.

The following example shows how to generate a default route if any of the routes that match 2001::/96 are present in the RIB:

```
route-policy ospfv3-originate
    if rib-has-route in (2001::/96) then
        pass
    endif
end-policy
```

```
router ospfv3 1
    default-information originate policy ospfv3-originate
    .
    .
    .
```

Redistribute

The redistribute attach point within OSPFv3 injects routes from other routing protocol sources into the OSPFv3 link-state database, which is done by selecting the route types it wants to import from each protocol. It then sets the OSPFv3 parameters of cost and metric type. The policy can control how the routes are injected into OSPFv3 by using the **metric type** command.

The following example shows how to redistribute routes from BGP instance instance_15 into OSPF instance 1 using the policy OSPFv3-redist. The policy sets the metric type to type-2 for all redistributed routes. BGP routes with a tag of 10 have their cost set to 100, and BGP routes with a tag of 20 have their OSPFv3 cost set to 200. Any BGP routes not carrying a tag of either 10 or 20 are not be redistributed into the OSPFv3 link-state database.

```
route-policy OSPFv3-redist
    set metric-type type-2
    if tag eq 10 then
        set cost 100
    elseif tag eq 20 then
        set cost 200
    else
        drop
    endif
end-policy

router ospfv3 1
    redistribute bgp instance_15 policy OSPFv3-redist
    .
    .
    .
```

OSPFv3 Attributes and Operators

Table 6 summarizes the OSPFv3 attributes and operators.

Table 6 OSPFv3 Attributes and Operators

Attribute	Match	Set
cost	n/a	set
destination	in	n/a
metric-type	n/a	set
rib-has-route	in	n/a
route-type	is	n/a
tag	eq, ge, le	set

IS-IS Policy Attach Points

This section describes each of the IS-IS policy attach points and provides a summary of the BGP attributes and operators.

Redistribute

The redistribute attach point within IS-IS allows routes from other protocols to be readvertised by IS-IS. The policy is a set of control structures for selecting the types of routes that a user wants to redistribute into IS-IS. The policy can also control which IS-IS level the routes are injected into and at what metric values.

The following example shows how to redistribute routes from IS-IS instance 1 into IS-IS instance instance_10 using the policy ISIS-redist. This policy sets the level to level-1-2 for all redistributed routes. OSPF routes with a tag of 10 have their metric set to 100, and IS-IS routes with a tag of 20 have their IS-IS metric set to 200. Any IS-IS routes not carrying a tag of either 10 or 20 are not be redistributed into the IS-IS database.

```
route-policy ISIS-redist
    set level level-1-2
    if tag eq 10 then
        set metric 100
    elseif tag eq 20 then
        set metric 200
    else
        drop
    endif
end-policy

router isis instance_10
    address-family ipv4 unicast
        redistribute ospf 1 policy ISIS-redist
        .
        .
        .
```

IS-IS Attributes and Operators

Table 7 summarizes the IS-IS attributes and operators.

Table 7 IS-IS Attributes and Operators

Attribute	Match	Set
Destination	in	n/a
Level	n/a	set
metric	n/a	set
metric-type	n/a	set
rib-has-route	in	n/a
route-type	is	n/a
tag	eq, ge, le	n/a

Attached Policy Modification

Policies that are in use do, on occasion, need to be modified. In the traditional configuration model, a policy modification would be done by completely removing the policy and re-entering it. However, this model allows for a window of time in which no policy is attached and default actions to be used, which is an opportunity for inconsistencies to exist. To close this window of opportunity, you can modify a policy in use at an attach point by respecifying it, which allows for policies that are in use to be changed, without having a window of time in which no policy is applied at the given attach point.

Note A route policy or set that is in use at an attach point cannot be removed because this removal would result in an undefined reference. An attempt to remove a route policy or set that is in use at an attach point results in an error message to the user.

Nonattached Policy Modification

As long as a given policy is not attached at an attach point, the policy is allowed to refer to nonexistent sets and policies. Configurations can be built that reference sets or policy blocks that are not yet defined, and then later those undefined policies and sets can be filled in. This method of building configurations gives much greater flexibility in policy definition. Every piece of policy you want to reference while defining a policy need not exist in the configuration. Thus, you can define a policy sample1 that references a policy sample2 using an apply statement even if the policy sample2 does not exist. Similarly, you can enter a policy statement that refers to a nonexistent set.

However, the existence of all referenced policies and sets is enforced when a policy is attached. Thus, if a user attempts to attach the policy sample1 with the reference to an undefined policy sample2 at an inbound BGP policy using the statement **neighbor 1.2.3.4 address-family ipv4 unicast policy sample1 in**, the configuration attempt is rejected because the policy sample2 does not exist.

Editing Routing Policy Configuration Elements

RPL is based on statements rather than on lines. That is, within the begin-end pair that brackets policy statements from the CLI, a new line is merely a separator, the same as a space character.

The CLI provides the means to enter and delete route policy statements. RPL provides a means to edit the contents of the policy between the begin-end brackets using a microemacs editor.

Editing Routing Policy Configuration Elements Using the EMACS Editor

To edit the contents of a routing policy, use the following CLI command in EXEC mode:

```
edit {route-policy | prefix-set | as-path-set | community-set | extended-community-set}
name
```

A copy of the route policy is copied to a temporary file and the editor is launched. After editing, save the changes by using the **save-buffer** command, C-X C-S (Control-X Control-S). To exit the editor, use the **quit** command, Control-X Control-C. When you quit the editor, the buffer is committed. If there are no parse errors, the configuration is committed:

```
RP/0/RP0/CPU0:router# edit route-policy policy_A
----------------------------------------
== MicroEMACS 3.8b () == rpl_edit.139281 ==
  if destination in (2001::/8) then
    drop
  endif
end-policy
!

== MicroEMACS 3.8b () == rpl_edit.139281 ==
Parsing.
83 bytes parsed in 1 sec (82)bytes/sec
Committing.
1 items committed in 1 sec (0)items/sec
Updating.
Updated Commit database in 1 sec

RP/0/RP0/CPU0:router#
```

If there are parse errors, you are asked whether editing should continue:

```
RP/0/RP0/CPU0:router#edit route-policy policy_B
== MicroEMACS 3.8b () == rpl_edit.141738
route-policy policy_B
 set metric-type type_1
 if destination in (2001::/8) then
    drop
  endif
end-policy
!
== MicroEMACS 3.8b () == rpl_edit.141738 ==
Parsing.
105 bytes parsed in 1 sec (103)bytes/sec

% Syntax/Authorization errors in one or more commands.!! CONFIGURATION
FAILED DUE TO SYNTAX/AUTHORIZATION ERRORS
 set metric-type type_1
 if destination in (2001::/8) then
    drop
  endif
end-policy
!

Continue editing? [no]:
```

If you answer **yes**, the editor continues on the text buffer from where you left off. If you answer **no**, the running configuration is not changed and the editing session is ended.

Editing Routing Policy Configuration Elements Using the CLI

The CLI allows you to enter and delete route policy statements. You can complete a policy configuration block by entering applicable commands such as **end-policy** or **end-set**. Alternatively, the CLI interpreter allows you to use the **exit** command to complete a policy configuration block. The **abort** command is used to discard the current policy configuration and return to global configuration mode.

How to Implement Routing Policy

This section contains the following procedures:

- Defining a Route Policy, page RC-237 (required)
- Attaching a Routing Policy to a BGP Neighbor, page RC-238 (required)
- Modifying a Routing Policy Using the Microemacs Editor, page RC-240 (optional)

Defining a Route Policy

This task explains how to define a route policy.

Note If you want to modify an existing routing policy using the command-line interface (CLI), you must redefine the policy by completing this task.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **end-policy**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`route-policy`** *`name`*
	Example:
	`RP/0/RP0/CPU0:router(config)# route-policy sample1`
		Enters route-policy configuration mode.
		• After the route-policy has been entered, a group of commands can be entered to define the route-policy.

	Command or Action	Purpose

Step 3	`end-policy`
	Example:
	`RP/0/RP0/CPU0:router(config-rpl)# end-policy`
		Ends the definition of a route policy and exits route-policy configuration mode.

Step 4	`end`
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Attaching a Routing Policy to a BGP Neighbor

This task explains how to attach a routing policy to a BGP neighbor. The procedure to attach a routing policy to an IS-IS or OSPF neighbor is the same as BGP, except that the commands and applicable arguments vary.

Prerequisites

A routing policy must be preconfigured and well defined prior to it being applied at an attach point. If a policy is not predefined, an error message is generated stating that the policy is not defined.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** ***ip-address***
4. **address-family** {**ipv4** | **ipv6**} {**multicast** | **unicast**}
5. **route-policy** *route-policy-name* {**in** | **out**}
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 125`	Configures a BGP routing process and enters router configuration mode. • The *as-number* argument identifies the autonomous system in which the router resides. Valid values are from 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.20`	Specifies a neighbor IP address.

Step 4	**address-family** {**ipv4** | **ipv6**} {**multicast** | **unicast**}
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
		Specifies the address family, the version of IP that is in use, and either multicast or unicast.
		• Enters address family configuration mode.

	Command or Action	Purpose

Step 5	**`route-policy`** *`policy-name`* `{`**`in`** `|` **`out`**`}`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy example1 in`
		Attaches the route-policy, which must be well formed and predefined.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Modifying a Routing Policy Using the Microemacs Editor

This task explains how to modify an existing routing policy using the microemacs editor.

SUMMARY STEPS

1. **edit** {**route-policy** | **prefix-set** | **as-path-set** | **community-set** | **extended-community-set**} *name*
2. **show rpl route-policy** *name* [**detail**]
3. **show rpl prefix-set** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**edit** {**route-policy** | **prefix-set** | **as-path-set** | **community-set** | **extended-community-set**} *name* **Example:** `RP/0/RP0/CPU0:router# edit route-policy sample1`	Identifies the route policy, prefix set, AS path set, community set, or extended community set name to be modified. • A copy of the route policy, prefix set, AS path set, community set, or extended community set is copied to a temporary file and the microemacs editor is launched. When you finish editing the policy or set, save the changes by using the **save-buffer** command, ^X^S (Control-X Control-S). To commit the changed configuration: • save the buffer (Control-X Control-S) • exit MicroEmacs (Control-X Control-C)
Step 2	**show rpl route-policy** *name* [**detail**] **Example:** `RP/0/RP0/CPU0:router# show rpl route-policy sample2`	(Optional) Displays the configuration of a specific named route policy. • Use the **detail** keyword to display all policies and sets that a policy uses.
Step 3	**show rpl prefix-set** *name* **Example:** `RP/0/RP0/CPU0:router# show rpl prefix-set prefixset1`	(Optional) Displays the contents of a named prefix set. • To display the contents of a named AS path set, community set, or extended community set, replace the **prefix-set** keyword with **as-path-set**, **community-set**, or **extcommunity-set**, respectively.

Configuration Examples for Implementing Routing Policy

This section provides the following configuration examples:

- Routing Policy Definition: Example, page RC-241
- Simple Inbound Policy: Example, page RC-242
- Modular Inbound Policy: Example, page RC-243
- Translating Cisco IOS Route Maps to Cisco IOS XR Routing Policy Language: Example, page RC-244

Routing Policy Definition: Example

In the following example, a BGP route policy named sample1 is defined using the **route-policy** *name* command. The policy compares the network layer reachability information (NLRI) to the elements in the prefix set test. If it evaluates to true, the policy performs the operations in the *then* clause. If it evaluates to false, the policy performs the operations in the *else* clause, that is, sets the MED value to 200 and adds the community 2:100 to the route. The final steps of the example commit the configuration to the router, exit configuration mode, and display the contents of route policy sample1.

```
configure
    route-policy sample1
       if destination in test then
         drop
       else
         set med 200
         set community (2:100) additive
       endif
    end-policy
    end
show config running route-policy sample1

Building configuration...
    route-policy sample1
       if destination in test then
         drop
       else
         set med 200
         set community (2:100) additive
       endif
    end-policy
```

Simple Inbound Policy: Example

The following policy discards any route whose network layer reachability information (NLRI) specifies a prefix longer than /24, and any route whose NLRI specifies a destination in the address space reserved by RFC 1918. For all remaining routes, it sets the MED and local preference, and adds a community to the list in the route.

For routes whose community lists include any values in the range from 101:202 to 106:202 that have a 16-bit tag portion containing the value 202, the policy prepends autonomous system number 2 twice, and adds the community 2:666 to the list in the route. Of these routes, if the MED is either 666 or 225, then the policy sets the origin of the route to incomplete, and otherwise sets the origin to IGP.

For routes whose community lists do not include any of the values in the range from 101:202 to 106:202, the policy adds the community 2:999 to the list in the route.

```
prefix-set too-specific
    0.0.0.0/0 ge 25 le 32
end-set

prefix-set rfc1918
    10.0.0.0/8 le 32,
    172.16.0.0/12 le 32,
    192.168.0.0/16 le 32
end-set

route-policy inbound-tx
    if destination in too-specific or destination in rfc1918 then
        drop
    endif
    set med 1000
    set local-preference 90
    set community (2:1001) additive
    if community matches-any ([101..106]:202) then
        prepend as-path 2 2
        set community (2:666) additive
        if med is 666 or med is 225 then
            set origin incomplete
        else
            set origin igp
        endif
    else
        set community (2:999) additive
    endif
end-policy

router bgp 2
    neighbor 10.0.1.2 address-family ipv4 unicast route-policy inbound-tx in
```

Modular Inbound Policy: Example

The following policy example shows how to build two inbound policies, in-100 and in-101, for two different peers. In building the specific policies for those peers, the policy reuses some common blocks of policy that may be common to multiple peers. It builds a few basic building blocks, the policies common-inbound, filter-bogons, and set-lpref-prepend.

The filter-bogons building block is a simple policy that filters all undesirable routes, such as those from the RFC 1918 address space. The policy set-lpref-prepend is a utility policy that can set the local preference and prepend the AS path according to parameterized values that are passed in. The common-inbound policy uses these filter-bogons building blocks to build a common block of inbound policy. The common-inbound policy is used as a building block in the construction of in-100 and in-101 along with the set-lpref-prepend building block.

This is a simple example that illustrates the modular capabilities of the policy language.

```
prefix-set bogon
  10.0.0.0/8 ge 8 le 32,
  0.0.0.0,
  0.0.0.0/0 ge 27 le 32,
  192.168.0.0/16 ge 16 le 32
end-set
!
route-policy in-100
  apply common-inbound
  if community matches-any ([100..120]:135) then
    apply set-lpref-prepend (100,100,2)
    set community (2:1234) additive
  else
    set local-preference 110
  endif
  if community matches-any ([100..666]:[100..999]) then
    set med 444
    set local-preference 200
    set community (no-export) additive
  endif
end-policy
!
route-policy in-101
  apply common-inbound
  if community matches-any ([101..200]:201) then
    apply set-lpref-prepend(100,101,2)
    set community (2:1234) additive
  else
    set local-preference 125
  endif
end-policy
!
route-policy filter-bogons
  if destination in bogon then
        drop
    else
```

```
        pass
  endif
end-policy
!
route-policy common-inbound
  apply filter-bogons
  set origin igp
  set community (2:333)
end-policy
!
route-policy set-lpref-prepend($lpref,$as,$prependcnt)
  set local-preference $lpref
  prepend as-path $as $prependcnt
end-policy
```

Translating Cisco IOS Route Maps to Cisco IOS XR Routing Policy Language: Example

RPL performs the same functions as route-maps. See the *Converting Cisco IOS Configurations to Cisco IOS XR Configurations* guide.

Additional References

The following sections provide references related to implementing RPL.

Related Documents

Related Topic	Document Title
Routing policy language commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Routing Policy Language Commands on Cisco IOS XR Software*, Release 3.2
Regular expression syntax	"Understanding Regular Expressions, Special Characters and Patterns" appendix in the *Cisco IOS XR Getting Started Guide*

Standards

Standards	Title
Draft-ietf-idr-bgp4-26.txt	*A Border Gateway Protocol 4*, by Y. Rekhter, T.Li, S. Hares

MIBs

MIBs	MIBs Link
There are no applicable MIBs for this module.	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Static Routes on Cisco IOS XR Software

Static routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the Cisco IOS XR software cannot build a route to a particular destination. They are useful for specifying a gateway of last resort to which all unroutable packets are sent.

This module describes the tasks you need to implement static routes on your Cisco IOS XR network.

Note For more information about static routes on the Cisco IOS XR software and complete descriptions of the static routes commands listed in this module, see the "Related Documents" section of this module. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing Static Routes on Cisco IOS XR Software

Release	Modification
Release 2.0	This feature was introduced on the Cisco CRS-1.
Release 3.0	No modification.
Release 3.2	Support was added for the Cisco XR 12000 Series Router.

Contents

Prerequisites for Implementing Static Routes on Cisco IOS XR Software

To use this command, you must be in a user group associated with a task group that includes the proper task IDs. For detailed information about user groups and task IDs, see the *Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*.

Information About Implementing Static Routes on Cisco IOS XR Software

To implement static routes you need to understand the following concepts:

- Static Route Functional Overview, page RC-248
- Default Administrative Distance, page RC-248
- Directly Connected Routes, page RC-249

Static Route Functional Overview

Static routes are entirely user configurable and can point to a next-hop interface, next-hop IP address, or both. In Cisco IOS XR software, if an interface was specified, then the static route is installed in the Routing Information Base (RIB) if the interface is reachable. If an interface was not specified, the route is installed if the next-hop address is reachable. The only exception to this configuration is when a static route is configured with the permanent attribute, in which case it is installed in RIB regardless of reachability.

Networking devices forward packets using route information that is either manually configured or dynamically learned using a routing protocol. Static routes are manually configured and define an explicit path between two networking devices. Unlike a dynamic routing protocol, static routes are not automatically updated and must be manually reconfigured if the network topology changes. The benefits of using static routes include security and resource efficiency. Static routes use less bandwidth than dynamic routing protocols, and no CPU cycles are used to calculate and communicate routes. The main disadvantage to using static routes is the lack of automatic reconfiguration if the network topology changes.

Static routes can be redistributed into dynamic routing protocols, but routes generated by dynamic routing protocols cannot be redistributed into the static routing table. No algorithm exists to prevent the configuration of routing loops that use static routes.

Static routes are useful for smaller networks with only one path to an outside network and to provide security for a larger network for certain types of traffic or links to other networks that need more control. In general, most networks use dynamic routing protocols to communicate between networking devices but may have one or two static routes configured for special cases.

Default Administrative Distance

Static routes have a default administrative distance of 1. A low number indicates a preferred route. By default, static routes are preferred to routes learned by routing protocols. Therefore, you can configure an administrative distance with a static route if you want the static route to be overridden by dynamic routes. For example, you could have routes installed by the Open Shortest Path First (OSPF) protocol with an administrative distance of 120. To have a static route that would be overridden by an OSPF dynamic route, specify an administrative distance greater than 120.

Directly Connected Routes

The routing table considers the static routes that point to an interface as "directly connected." Directly connected networks are advertised by IGP routing protocols if a corresponding **interface** command is contained under the router configuration stanza of that protocol.

In directly attached static routes, only the output interface is specified. The destination is assumed to be directly attached to this interface, so the packet destination is used as the next hop address. The following example shows how to specify that all destinations with address prefix 2001:0DB8::/32 are directly reachable through interface GigabitEthernet 0/5/0/0:

```
RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 gigabitethernet 0/5/0/0
```

Directly attached static routes are candidates for insertion in the routing table only if they refer to a valid interface; that is, an interface that is both up and has IPv4 or IPv6 enabled on it.

Recursive Static Routes

In a recursive static route, only the next hop is specified. The output interface is derived from the next hop. The following example shows how to specify that all destinations with address prefix 2001:0DB8::/32 are reachable through the host with address 2001:0DB8:3000::1:

```
RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1
```

A recursive static route is valid (that is, it is a candidate for insertion in the routing table) only when the specified next hop resolves, either directly or indirectly, to a valid output interface, provided the route does not self-recurse, and the recursion depth does not exceed the maximum IPv6 forwarding recursion depth.

A route self-recurses if it is itself used to resolve its own next hop. If a static route becomes self-recursive, RIB sends a notification to static routes to withdraw the recursive route.

The following example shows how to define a recursive IPv6 static route:

```
RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1
```

This static route is not inserted into the IPv6 routing table because it is self-recursive. The next hop of the static route, 2001:0DB8:3000:1, resolves through the BGP route 2001:0DB8:3000:0/16, which is itself a recursive route (that is, it only specifies a next hop). The next hop of the BGP route, 2001:0DB8::0104, resolves through the static route. Therefore, the static route would be used to resolve its own next hop.

It is not normally useful to manually configure a self-recursive static route, although it is not prohibited. However, a recursive static route that has been inserted in the routing table may become self-recursive as a result of some transient change in the network learned through a dynamic routing protocol. If this occurs, the fact that the static route has become self-recursive will be detected and it will be removed from the routing table, although not from the configuration. A subsequent network change may cause the static route to no longer be self-recursive, in which case it will be re-inserted in the routing table.

Fully Specified Static Routes

In a fully specified static route, both the output interface and next hop are specified. This form of static route is used when the output interface is a multiaccess one and it is necessary to explicitly identify the next hop. The next hop must be directly attached to the specified output interface. The following example shows a definition of a fully specified static route:

```
RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1
```

A fully specified route is valid (that is, a candidate for insertion into the routing table) when the specified interface is IPv4 or IPv6 enabled and up.

Floating Static Routes

Floating static routes are static routes that are used to back up dynamic routes learned through configured routing protocols. A floating static route is configured with a higher administrative distance than the dynamic routing protocol it is backing up. As a result, the dynamic route learned through the routing protocol is always preferred to the floating static route. If the dynamic route learned through the routing protocol is lost, the floating static route is used in its place. The following example shows how to define a floating static route:

```
RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1 210
```

Any of the three types of static routes can be used as a floating static route. A floating static route must be configured with an administrative distance that is greater than the administrative distance of the dynamic routing protocol because routes with smaller administrative distances are preferred.

Note By default, static routes have smaller administrative distances than dynamic routes, so static routes preferred to dynamic routes.

How to Implement Static Routes on Cisco IOS XR Software

This section contains the following procedures:

- Configuring a Static Route, page RC-250 (required)
- Configuring a Floating Static Route, page RC-251 (optional)
- Changing the Maximum Number of Allowable Static Routes, page RC-253 (optional)

Configuring a Static Route

This task explains how to configure a static route.

SUMMARY STEPS

1. **configure**
2. **route** {**ipv4** | **ipv6**} {**unicast** | **multicast**} *prefix mask* {*ip-address* | *interface-type interface-instance*} [*distance*] [**tag** *tag*] [**permanent**]

3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`route`** `{`**`ipv4`** `|` **`ipv6`**`}` `{`**`unicast`** `|` **`multicast`**`}` *`prefix mask`* `{`*`ip-address`* `|` *`interface-type interface-instance`*`}` `[`*`distance`*`]` `[`**`tag`** *`tag`*`]` `[`**`permanent`**`]`
	Example:
	`RP/0/RP0/CPU0:router(config)# route ipv4 unicast 10.0.0.0/8 172.20.16.6 110`
		Configures an administrative distance of 110.
		• This example shows how to route packets for network 10.0.0.0 through to a router at 172.20.16.6 if dynamic information with administrative distance less than 110 is not available.

Step 3	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Floating Static Route

This task explains how to configure a floating static route.

SUMMARY STEPS

1. **configure**
2. **route** {**ipv4** | **ipv6**} {**unicast** | **multicast**} *prefix mask* {*ip-address* | *interface-type interface-instance*} [*distance*] [**tag** *tag*] [**permanent**]

3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/RP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`route`** `{`**`ipv4`** `|` **`ipv6`**`}` `{`**`unicast`** `|` **`multicast`**`}` *`prefix mask`* `{`*`ip-address`* `|` *`interface-type interface-instance`*`}` `[`*`distance`*`]` `[`**`tag`** *`tag`*`]` `[`**`permanent`**`]`
	Example:
	`RP/0/RP0/CPU0:router(config)# route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1 201`
		In this example, a floating static IPv6 route is being configured. An administrative distance of 201 is configured
		Default administrative distances are as follows:
		• Connected interface—0
		• Static route—1

Step 3	**`end`**
	or
	`commit`
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Changing the Maximum Number of Allowable Static Routes

This task explains how to change the maximum number of allowable static routes.

Restrictions

The number of static routes that can be configured on a router for a given address family is limited by default to 4000. The limit can be raised or lowered using the **route maximum** command. Note that if you use the **route maximum** command to reduce the configured maximum allowed number of static routes for a given address family below the number of static routes currently configured, the change is rejected. In addition, understand the following behavior: If you commit a batch of routes that would, when grouped, push the number of static routes configured above the maximum allowed, the first *n* routes in the batch are accepted. The number previously configured is accepted, and the remainder are rejected. The *n* argument is the difference between the maximum number allowed and number previously configured.

SUMMARY STEPS

1. **configure**
2. **route maximum** {**ipv4** | **ipv6**} *value*
3. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route maximum** {**ipv4** | **ipv6**} *value* **Example:** `RP/0/RP0/CPU0:router(config)# route maximum ipv4 10000`	Changes the maximum number of allowable static routes. • Specify IPv4 or IPv6 address prefixes. • Specify the maximum number of static routes for the given address family. The range is from 1 to 128000. • This example sets the maximum number of static IPv4 routes to 10000.

Step 3	**end**
	or
	commit
	Example:
	`RP/0/RP0/CPU0:router(config)# end`
	or
	`RP/0/RP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples

This section provides the following configuration examples:

- Configuring Traffic Discard: Example
- Configuring a Fixed Default Route: Example
- Configuring a Floating Static Route: Example

Configuring Traffic Discard: Example

Configuring a static route to point at interface null 0 may be used for discarding traffic to a particular prefix. For example, if it is required to discard all traffic to prefix 2001:0DB8:42:1/64, the following static route would be defined:

```
configure
 route ipv6 unicast 2001:0DB8:42:1::/64 null 0
 end
```

Configuring a Fixed Default Route: Example

A default static route is often used in simple router topologies. In the following example, a router is configured with an administrative distance of 110.

```
configure
 route ipv4 unicast 10.0.0.0/8 172.20.16.6 110
 end
```

Configuring a Floating Static Route: Example

A floating static route often is used to provide a backup path if connectivity fails. In the following example, a router is configured with an administrative distance of 201.

```
configure
 route ipv6 unicast 2001:0DB8::/32 2001:0DB8:3000::1 201
 end
```

Where to Go Next

For additional information on static routes, routing protocols, and RIB, consult the following publications:

- *Implementing and Monitoring RIB on Cisco IOS XR Software*
- *Implementing BGP on Cisco IOS XR Software*
- *Implementing IS-IS on Cisco IOS XR Software*
- *Implementing OSPF on Cisco IOS XR Software*
- *Implementing OSPFv3 on Cisco IOS XR Software*
- *RIB Commands on Cisco IOS XR Software*

Additional References

The following sections provide references related to implementing static routes on Cisco IOS XR software.

Related Documents

Related Topic	Document Title
Static routes commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Static and Utility Routing Commands on Cisco IOS XR Software*, Release 3.2

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
There are no applicable MIBs for this module.	To locate and download MIBs for selected platforms using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

INDEX

HC	Cisco IOS XR Interface and Hardware Component Configuration Guide
IC	Cisco IOS XR IP Addresses and Services Configuration Guide
MCC	Cisco IOS XR Multicast Configuration Guide
MPC	Cisco IOS XR MPLS Configuration Guide
QC	Cisco IOS XR Modular Quality of Service Configuration Guide
RC	Cisco IOS XR Routing Configuration Guide
SC	Cisco IOS XR System Security Configuration Guide
SMC	Cisco IOS XR System Management Configuration Guide